Constructing a Functional Quadcopter: Control Theory, Sensor Fusion, and Engineering Challenges

Project in Electrical engineering 1TE675

Jonas Gartner
Sigge Axelsson

January 16, 2025

Abstract

This report outlines the design and construction of a quadcopter drone with a focus on implementing advanced control theory, signal processing, and sensor fusion techniques. The primary objective was to develop a functional drone capable of reliable and sustained flight using custom-built hardware and a robust flight controller.

The system incorporates a cascaded PID controller for attitude stabilization and a Kalman filter for real-time state estimation, integrating gyroscopic and accelerometer data. Additionally, an altitude measurement system utilizing a combination of Kalman filtering and zero-velocity updates was explored in an attempt to improve height estimation. The drone's hardware includes an ESP32-C3 microcontroller, Brushless DC-motors electronic speed controllers, MPU6050 inertial sensor, and a BNP280 pressure sensor, along with critical components mounted on an circuit board. A custom remote control was also constructed from scratch.

The quadcopter demonstrated stable hover and basic maneuvering capabilities, meeting key project objectives. Challenges included time constraints that limited the full implementation of the altitude control system. This project highlights the effectiveness of cascaded control and Kalman-based estimation in quadcopter design while providing a foundation for future advancements in control optimization and sensor fusion techniques.

Acknowledgements

We would like to extend our gratitude to Carbon Aeronautics, whose report and shared resources provided valuable guidance throughout our project. Their work significantly influenced some aspects of our design and implementation, and we are grateful for their contribution to this field.

Contents

1 Introduction

1.1 Background

Unmanned aerial vehicles (UAVs), commonly known as drones, have become indispensable tools across various industries due to their ability to perform tasks that are challenging, hazardous, or inefficient for humans. Among these, quadcopter drones are particularly notable for their compact design and precise control over multiple degrees of freedom. This versatility makes them ideal for a wide range of applications, including aerial photography, delivery systems, and scientific exploration.

The design and development of a quadcopter involve a multidisciplinary approach that integrates control theory, embedded systems, and signal processing. This project aims to advance the field by demonstrating practical engineering solutions using affordable, off-the-shelf components to achieve stable and reliable flight in a custom-built quadcopter.

1.2 Purpose

The purpose of this project is to design and build a fully functional quadcopter from the ground up. The focus lies on applying control theory and signal processing on a microcontroller, alongside leveraging fundamental electronics and CAD skills to design and construct the necessary hardware. The project seeks to provide a hands-on exploration of the challenges and solutions involved in creating a custom UAV.

1.3 Project Specifications

1. Quadcopter should contain a wireless battery pack with an integrated Battery Management System (BMS) for safe and efficient power management.
2. All electronic components are designed from the ground up to ensure customization and optimization for the project's specific requirements.
3. Custom PCBs tailored to the system's specifications are designed
4. An advanced flight controller that enhances stability and responsiveness for precise control is implemented.
5. Reliable, sustained flight capability with robust handling of flight dynamics is achieved.

2 Theory

2.1 PID Controller

A PID controller is a control algorithm used to maintain a desired output by adjusting inputs based on three components: Proportional (P), Integral (I), and Derivative (D).

The proportional part corrects the error based on the current difference between desired and actual values. The integrating part accounts for past errors to eliminate any residual offset. The derivative part predicts future errors by considering the rate of change. Together, these components help achieve stable, precise, and responsive control in systems like temperature regulation or motor control.

Below are the equations describing the PID feedback of the quadcopter. Equation 1 describes a general PID controller in continuous time, and equation 2 describes a PID controller in discrete time:

$$Input_{\text{motor}}(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{1}$$

$$Input_{\text{motor}}(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i)\Delta t + K_d \frac{e(k) - e(k-1)}{\Delta t} \tag{2}$$

2.2 State Space Models

A state-space model is a mathematical framework used to describe and analyze dynamic systems in terms of their internal state variables and external inputs. It is commonly used in control systems, signal processing, and systems engineering. A state space model commonly consists of a state equation (3) and a measurements or output equation (4). Observe that this specific state space model is in discrete form.

$$x_{k+1} = Fx_k + Gu_k + w_k \tag{3}$$

$$y_k = Hx_k + v_k \tag{4}$$

- x_k: The *state vector* at time step k. It represents the internal state of the system. A state can be defined as any information thats relevant for the system i.e speed, temperature or weight.

- F: The *state transition matrix*. It governs how the state evolves from k to $k+1$ in the absence of inputs and disturbances.
- G: The *input (or control) matrix*. It describes how the control input u_k influences the state.
- u_k: The *control input vector* at time step k. It represents external inputs to the system.
- w_k: The *process noise vector*. It models uncertainties or random disturbances affecting the system's evolution.
- y_k: The *output (or measurement) vector* at time step k. It represents observable quantities from the system.
- H: The *output (or observation) matrix*. It maps the system's state x_k to its output y_k.
- v_k: The *measurement noise vector*. It represents uncertainties or noise in the observed measurements.

2.2.1 Discretizing Continuous State Space Models

A continous state space model can be represented as 5 and 6 while its discrete equivalent is represented as 7 and 8. The necessary coefficients are calculated according to 12, 10, 10 and 11. [1]

$$\dot{x}(t) = A * x(t) + Bu(t) \tag{5}$$

$$y(t) = C * x(t) \tag{6}$$

$$\dot{x}(t) = F * x(t) + G * u(t) \tag{7}$$

$$y(t) = H * x(t) \tag{8}$$

$$F = e^{A*T_s} = \sum_{k=0}^{\infty} \frac{1}{k!}(A * T_s)^k \tag{9}$$

$$G = (e^{AT_s} - I)A^{-1}B \tag{10}$$

$$H = C \tag{11}$$

$$J = D \tag{12}$$

2.3 Cascaded Controllers

Cascade control is a multi-loop control architecture widely used in process industries to enhance control performance in the presence of unmeasurable disturbances and improve single-loop control effectiveness. In this configuration, a primary controller regulates the main process, while a secondary controller manages a supporting process. [2]

A major advantage of cascade control is its ability to address disturbances in the secondary loop quickly, ensuring they are corrected before impacting the primary process. The primary controller's output is used to adjust the setpoint for the secondary controller, which then generates the necessary control signal for effective regulation of the process. [2]

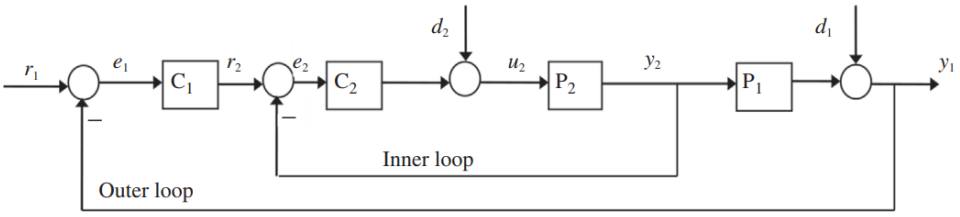


Figure 1: Cascade controller

It has also been shown that this type of controller is especially suitable for quadcopter drones. [3].

2.4 Quadcopter Dynamics

A quadcopter is a type of helicopter with four rotors, each generating lift independently. It is an under-actuated system with four input forces—one for each rotor—and six degrees of freedom (6DOF). Unlike traditional helicopters, which use variable-pitch rotors to control movement, a quadcopter relies on fixed-pitch and fixed-angle rotors.

Control in 6DOF is achieved by varying the speed (RPM) of each rotor individually. This variation adjusts the lift and rotational forces, allowing the quadcopter to tilt, or "roll" and "pitch," toward the direction of the slower-spinning rotor. These tilting motions redirect thrust into different directions, enabling linear movement.

Yaw control is achieved by pairing rotors to spin in opposite directions (clockwise and counterclockwise), which balances the drag-induced torque. The center of gravity is usually aligned with the plane of the rotors. However, minor efficiency differences between motors, even of the same type, can make stabilization more challenging.

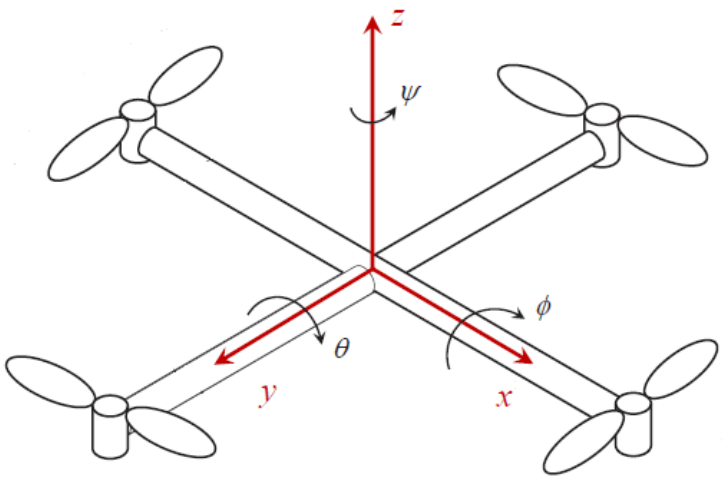


Figure 2: Roll (ϕ), Pitch (θ) and Yaw (ψ) are terms commonly used to describe rotation around the horizontal x axis, horizontal y axis and vertical z axis.

Unlike traditional helicopters, quadcopters require precise control systems to maintain balance and stability, as manual stabilization is extremely difficult. These systems are essential for achieving smooth and controlled flight.

2.4.1 Quadcopter State Space Model

A common 6 Degrees of freedom state space quadcopter model can be seen below [4]. In this example the state space model is given in continous form, meaning x_{k+1} is denoted as x' while $\dot{x}_{k+1}$ is denoted as x''.

$$\begin{bmatrix} x' \\ y' \\ z' \\ x'' \\ y'' \\ z'' \\ \phi' \\ \theta' \\ \psi' \\ \phi'' \\ \theta'' \\ \psi'' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ x' \\ y' \\ z' \\ \phi \\ \theta \\ \psi \\ \phi' \\ \theta' \\ \psi' \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ x' \\ y' \\ z' \\ \phi \\ \theta \\ \psi \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{14}$$

Note: the input vector u is defined as follows

- U_1 Total upward force along z axis
- U_2 pitch torque (about x axis)
- U_3 roll torque (about y axis)
- U_4 yaw torque (about z axis)

2.4.2 Simplified Quadcopter Dynamics

A simplified 3 degrees of freedom state space model of quadcopter dynamics can be seen below [4]. As can be seen in the state vector this model only takes angle and angular velocity into account. This can be used to balance a quadcopter and make it move around, the speed will however be unknown to the controller and would have to be controlled manually by a pilot.

$$\begin{bmatrix} \phi' \\ \theta' \\ \psi' \\ \phi'' \\ \theta'' \\ \psi'' \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \\ \phi' \\ \theta' \\ \psi' \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{d}{I_x} & 0 & -\frac{d}{I_x} \\ \frac{d}{I_y} & 0 & -\frac{d}{I_y} & 0 \\ -\frac{c}{I_z} & \frac{c}{I_z} & -\frac{c}{I_z} & \frac{c}{I_z} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \\ \phi' \\ \theta' \\ \psi' \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{16}$$

Note: The input vector is defined as follows.

- F_1 upwards force from motor 1
- F_2 upwards force from motor 2
- F_3 upwards force from motor 3
- F_4 upwards force from motor 4

2.5 Kalman Filter

[5] [6] The Kalman filter is a Linear Minimum mean square error (LMMSE) estimator which can estimate the states of a state space model. The notation $\hat{x}_{k|k-1}$ means that we estimate x at time k based on the information available at time k-1. $\hat{\theta}_{k|k} = \hat{\theta}_{k|k-1} + K_k(y_k - \hat{\theta}_{k|k-1})$ The kalman equations are the following.

Predictions

$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Gu_{k-1}$ (LMMSE prediction of x_k)
$P_{k|k-1} = FP_{k-1|k-1}F^\top + Q$ (error covariance associated with the LMMSE prediction $\hat{x}_{k|k-1}$)

note: *Q is the autocovariance of the process noise*

Updates

$$
\begin{aligned}
\tilde{y}_k &= y_k - H\hat{x}_{k|k-1} && \text{(Innovation - Difference between model \& observation)} \\
S_k &= HP_{k|k-1}H^\top + R && \text{(Covariance of innovation, } \mathbb{E}[(y_k - H\hat{x}_{k|k-1})(y_k - H\hat{x}_{k|k-1})^\top]) \\
K_k &= P_{k|k-1}H^\top S_k^{-1} && \text{(Optimal Kalman gain)} \\
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k\tilde{y}_k && \text{(LMMSE filter estimate of } x) \\
P_{k|k} &= P_{k|k-1} - K_kHP_{k|k-1} && \text{(error covariance associated with the LMMSE estimate } \hat{x}_{k|k})
\end{aligned}
$$

note: *R is the autocovariance of the measurement noise*

2.6 Zero Velocity Measurement

[7] [8] Zero Velocity Update (ZUPT) is a popular technique in inertial navigation systems for correction of sensor drift and improving the accuracy of position and velocity estimates. It relies on identifying periods of stationarity, during which the system's velocity is assumed to be zero, to introduce so called pseudo-measurements into the state-space model. These pseudo-measurements serve as corrections, helping to counteract cumulative errors such as accelerometer bias and integration drift, inherent in low-cost inertial measurement units (IMU).

State Transition and Measurement Model

[7] The state transition model describes how the state evolves from one time step to the next based on the system's current state, control input, and process noise. It is given by the following equation:

$$x_{k+1} = f(x_k, u_k) + w_k \tag{17}$$

where:

- x_k: State vector, which includes the system's position, velocity, and orientation (among other states).
- u_k: Control input, which typically includes accelerations measured by the IMU.
- w_k: Process noise, assumed to be Gaussian with covariance Q_k, representing random uncertainties in the system.

Here, x_k represents the state vector, which includes the system's position, velocity, and orientation (among other states). The term u_k denotes the control input, which

typically includes accelerations measured by the IMU. The process noise, w_k, is assumed to be Gaussian with covariance Q_k and represents random uncertainties in the system.

[7]The measurement model relates the system's state to observable quantities, which can be measured by sensors like the accelerometer or barometer. In the case of ZUPT, the measurement model introduces pseudo-measurements of zero velocity during stationary periods:

$$y_k = h(x_k) + e_k, \quad \text{if } D_{\theta,k} = 1 \tag{18}$$

where:

- y_k: Measurement (pseudo-measurement) at time step k, representing zero velocity.
- $h(x_k)$: Measurement function that models how the state x_k (specifically the velocity state) relates to the measured quantities.
- e_k: Measurement noise, assumed to be Gaussian with covariance R_k.
- $D_{\theta,k}$: Zero-velocity detector output, which is set to 1 when the system is stationary.

When zero velocity is detected (by the detection framework described in section 2.7.1), $D_{\theta,k}$ is set to 1 and the system is considered stationary. This sets the pseudo-measurement y_k to zero, which forces the system's velocity state, as represented by $h(x_k)$, to match the condition that the velocity is zero during stationary periods. As a result, the state vector is updated with this additional information, driving the velocity state to zero, correcting any drift that may have occurred.

2.7 Zero Velocity Detection

[7] [8] Zero velocity detection identifies moments when a system is stationary, setting $D_{\theta,k} = 1$ and applies the ZUPT. [8] Various detection methods exist for detecting zero velocity such as the Acceleration-Moving Variance Detector, Acceleration-Magnitude Detector and Angular Rate Energy Detector. In this report a Magnitude detection method has been studied and applied for the altitude estimation algorithm.

2.7.1 Acceleration-Moving Variance Detector

[8] This approach evaluates the variance of accelerometer measurements within a predefined time window to detect zero velocity. A low variance in the accelera-

tion data suggests that the system is stationary, allowing the assumption of zero velocity.

For a time window of N, zero velocity is detected if:

$$\frac{1}{N}\sum_{k=1}^{N}\left(\|a_k\| - \overline{\|a\|}\right)^2 < \gamma \tag{19}$$

where a_k is the acceleration vector of the altitude (direction of gravity) at time k, $\overline{\|a\|}$ is the mean acceleration magnitude over the window, and γ is a predefined variance threshold.

The variance-based approach is particularly effective in filtering out transient accelerations and noise, as it leverages the consistency of stationary conditions to infer zero velocity.

2.7.2 Statistical Background

[7] [8] Magnitude detection is a special case of hypothesis testing in statistical detection theory. The hypotheses are:

$$H_0 : \text{The sensor is in motion}, \quad H_1 : \text{The sensor is stationary}.$$

Under H_1, the measured acceleration magnitude should match g within noise limits. The likelihood ratio test (LRT) evaluates the probability of observing the data under each hypothesis:

$$L(z) = \frac{p(z \mid H_1)}{p(z \mid H_0)} \tag{20}$$

[8] where z represents the set of measurements. Detection is based on a threshold η, such that H_1 is chosen if: $L(z) > \eta$. These thresholds balance detection sensitivity and false alarm rates, influenced by environmental noise and sensor precision. Magnitude detection works well in structured conditions but can be affected by dynamic environments or high noise levels.

2.8 IMU Sensor Model

[9] Inertial measurement units (IMU) are widely used in navigation and robotics for precise motion tracking. However, their accuracy is limited by systematic errors inherent in the hardware. These errors stem from sensor misalignments, scaling

inaccuracies, and biases, which must be accounted for to obtain reliable measurements. The IMU sensor model uses mathematical representations to correct these errors, including misalignment matrices, scaling matrices, and bias vectors.

The corrected measurements for accelerometers and gyroscopes are modeled as follows:

$$a_O = T_a K_a (a_S + b_a + \nu_a) \tag{21}$$

$$\omega_O = T_g K_g (\omega_S + b_g + \nu_g) \tag{22}$$

where:

- a_O, ω_O: Corrected acceleration and angular velocity (in the orthogonal reference frame).
- a_S, ω_S: Raw sensor measurements (in the non-orthogonal sensor frame).
- b_a, b_g: Bias vectors accounting for constant offsets in measurements.
- ν_a, ν_g: Measurement noise vectors.
- T_a, T_g: Misalignment matrices to correct for non-orthogonal sensor axes.
- K_a, K_g: Scaling matrices to adjust for inaccuracies in axis-specific sensitivity.

Misalignment Matrices

IMUs often exhibit non-orthogonality between sensor axes due to manufacturing imperfections. The misalignment matrices T_a and T_g correct these errors by transforming the raw sensor data into an orthogonal reference frame:

$$T_a = \begin{bmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{bmatrix}, \quad T_g = \begin{bmatrix} 1 & -\gamma_{yz} & \gamma_{zy} \\ \gamma_{xz} & 1 & -\gamma_{zx} \\ -\gamma_{xy} & \gamma_{yx} & 1 \end{bmatrix} \tag{23}$$

Here, the parameters α_{ij} and γ_{ij} represent small angular misalignments between axes. These are calibrated during the correction process.

Scaling Matrices

Scaling inaccuracies occur when the sensitivity of each axis deviates from its nominal value. The scaling matrices K_a and K_g compensate for these deviations:

$$K_a = \begin{bmatrix} s_{ax} & 0 & 0 \\ 0 & s_{ay} & 0 \\ 0 & 0 & s_{az} \end{bmatrix}, \quad K_g = \begin{bmatrix} s_{gx} & 0 & 0 \\ 0 & s_{gy} & 0 \\ 0 & 0 & s_{gz} \end{bmatrix} \tag{24}$$

where s_{ax}, s_{ay}, s_{az}, s_{gx}, s_{gy}, and s_{gz} are the scale factors for the respective axes.

Bias Vectors

Biases represent constant offsets that are present in the sensor readings even when no external force or rotation is applied. These offsets are characterized by the bias vectors:

$$b_a = \begin{bmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{bmatrix}, \quad b_g = \begin{bmatrix} b_{gx} \\ b_{gy} \\ b_{gz} \end{bmatrix} \tag{25}$$

where b_{ax}, b_{ay}, b_{az} are accelerometer biases, and b_{gx}, b_{gy}, b_{gz} are gyroscope biases.

3 Implementation

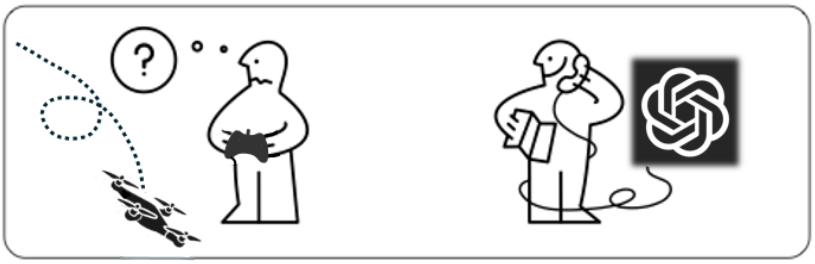


Figure 3: Method used for the implementation

3.1 Hardware and Components

This section gives a overview of the hardware and software used to construct and control the quadcopter drone.

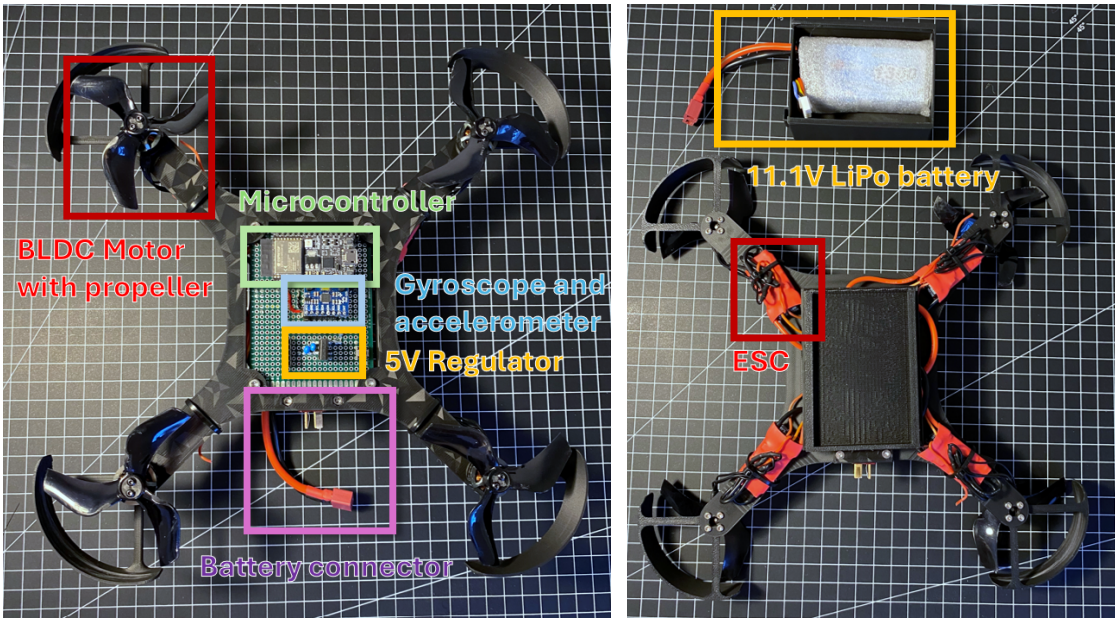
3.1.1 Quadcopter Components

Figure 4 together with the following list gives a overview of the components that are part of the quadcopter drone:

- ESP32C3 (DevKit02) Microcontroller with integrated wireless capabilities
- 1404 BLDC racing motors with 3" propellers
- MPU6050 Gyroscope and accelerometer
- 1300mAh 3S 39A 101g LiPo Battery (wrapped in bubble wrap).
- 5V Linear regulator with two 1uF capacitors
- 12A 2-4S PWM controlled brushless electronic speed controller

The microcontroller, sensor and voltage regulator are mounted on a experimental PCB as can be seen in figure 4a. The PCB and the electronic speed controllers (ESC) are connected to the battery with wago 221-415 clips that are located between the PCB and the black bottom panel in figure 4b. A complete list of the components used can be found on project GitHub page [1].

[1]Project github page: `https://github.com/ElektroJonas/DIY-Quadcopter/tree/main`



(a) Top (b) Bottom

Figure 4: Components of the quadcopter

3.1.2 Quadcopter Mechanical Design

The mechanical design of the quadcopter, shown in Figure 5, was developed in Onshape using an iterative design process. The primary focus was on optimizing the drone's robustness while minimizing its weight. The structure consists of a main frame, a battery enclosure, and propeller guards. The final design weighs 92 grams, though this could be reduced by removing the propeller guards and securing the battery with zip ties instead of using a dedicated enclosure. However, for this project, improved robustness was prioritized over slightly improved flight performance.

Figure 5: Mechanical design of quadcopter

3.1.3 Remote

To control the drone, a custom-designed remote was developed. The remote features a PCB created in KiCad and uses the same ESP32-C3 DevKit02 microcontroller as the drone. This choice enables the use of the ESP-NOW wireless protocol, allowing seamless communication between the remote and the drone without requiring external transceivers or receivers. To improve the user experience a grip for the remote was designed in Onshape and 3D printed. The remote can be seen in figure 6.

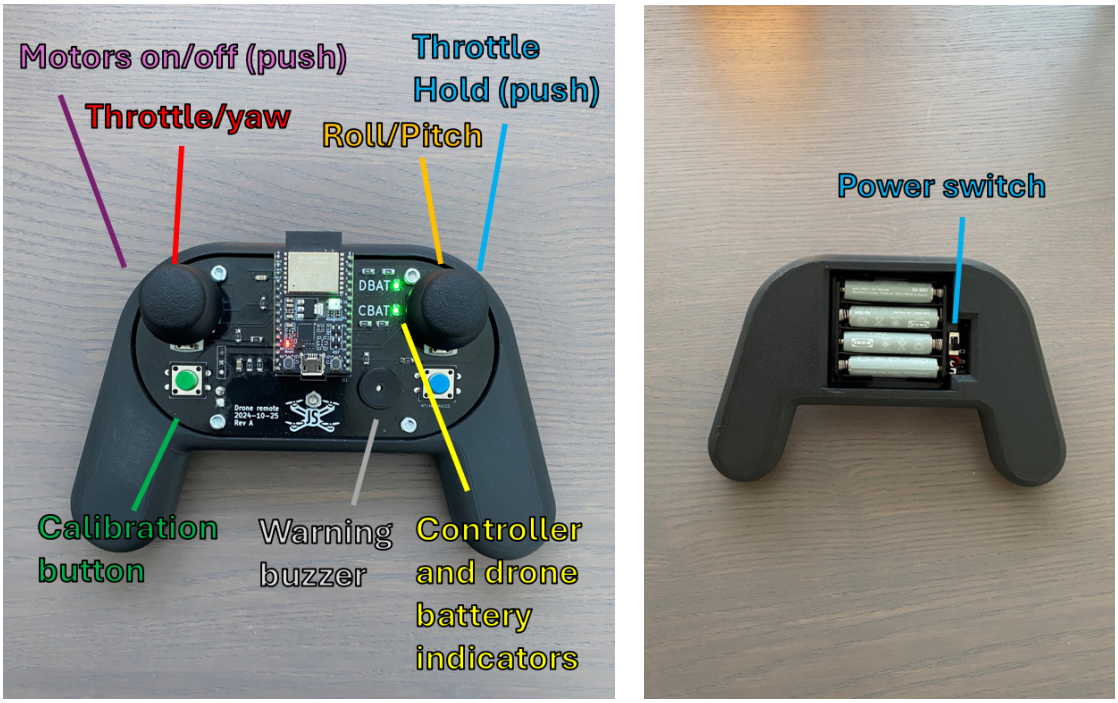


Figure 6: Drone remote and its features.

3.1.4 Battery

To power the quadcopter, a battery is required. The motors are designed to operate at 9.3–12.6V and draw a current of 12A each. Therefore, a battery with a high power density is essential. Three different battery packs were tested.

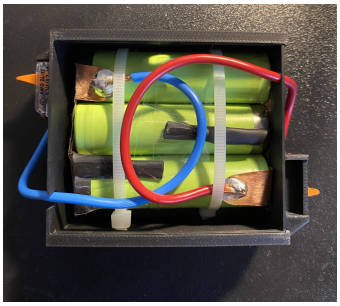
The first battery pack, shown in Figure 7a, consisted of three series-connected 21700 cylindrical lithium-ion cells, joined together using spot-welded copper. This configuration provided sufficient current and offered the highest energy density among the candidates. However, it was too heavy for the 1404 motors and 3" propellers, resulting in very poor performance.

The second candidate, shown in Figure 7b, was a pack of three series-connected 18650 cells, also connected by spot welding. These cells were significantly lighter but had a low continuous current rating, leading to voltage sag. This voltage sag causes ESC failure, resulting in violent crashes.

The third and final candidate, shown in Figure 7c, was a 3S 1300mAh pouch-cell LiPo battery, specifically designed for hobby drones. This battery was both lightweight and capable of delivering the required current, resulting in good flight performance. However, the downside of this battery is its low capacity, providing

only about 4 minutes of flight time. This candidate was chosen for the final build as it was the only one that has sufficient performance.

(a) 21700 3S battery pack
Capacity: 4000mAh
Max current: 40A
Weight: 210g



(b) 18650 3S battery pack
Capacity: 2000mAh
Max current: 30A
Weight: 135g

(c) Pouch 3S LiPo battery
Capacity: 1300mAh
Max current: 39A
Weight: 101

Figure 7: Different battery solutions that were tested with the quadcopter

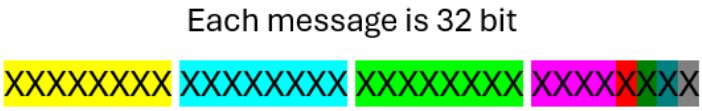
3.2 Wireless Communication

The wireless communication is based on the ESP-NOW protocol, supported by the ESP32 microcontroller series. ESP-NOW was selected for its ability to enable seamless communication between the remote and the quadcopter using the same type of microcontroller, without the need for additional external transmitters or receivers. Furthermore, it is easy to implement and provides a reported range of up to 200 meters under ideal conditions.

The protocol incorporates a Frame Check Sequence, an error-detection mechanism. This ensures that any message corrupted during transmission is discarded by the receiver, minimizing the risk of unintended behaviour caused by communication errors.

To enable the quadcopter to interpret messages from the remote, the messages must adhere to a predefined format, referred to as a communication protocol. Figure 8 illustrates the protocol design for this project.

The remote transmits a 32-bit message to the quadcopter every 20 ms, while the quadcopter sends an 8-bit message back to the remote every 100 ms. The 8-bit message contains a battery voltage reading, obtained using a voltage divider and the microcontroller's integrated 12-bit ADC. To fit the voltage value into an 8-bit message, the 4 least significant bits (LSBs) are discarded. The complete code for



Variable	Value	Description
Throttle	0-255	Sets common throttle term to make drone go up or down
Ref_x	0-255	Sets reference for rotating round x axis
Ref_y	0-255	Sets reference for rotating round y axis
Ref_z	0-7	Sets reference for rotating round z axis
Calibration	True/false	When true the drone restarts and calibrates the sensor during setup
MotorsOn	True/false	Turns on or off the motors
hold	True/false	PCOM will not change while hold is true
Unused	True/false	

Figure 8: Communication protocol that defines how wireless messages are interpreted by the quadcopter.

the remote can be found on GitHub. [2]

3.3 Maneuvering the Quadcopter

The strategy for maneuvering the quadcopter is illustrated in Figure 9. The power command to each of the four motors consists of a common term, $PCOM$, which is adjusted to control the quadcopter's vertical motion (up or down), and three differential terms, P_{regx}, P_{regy}, and P_{regz}. These differential terms allow the quadcopter to tilt along the horizontal x-axis (roll), tilt along the horizontal y-axis (pitch), or rotate about the vertical z-axis (yaw). Each differential term is managed by

[2]Project github page: `https://github.com/ElektroJonas/DIY-Quadcopter/tree/main`

individual controllers, which are described in detail in Section 3.4. The principles of roll and pitch control are shown in Figure 10.

Yaw control operates by reducing the speed of motors 1 and 3 while simultaneously increasing the speed of motors 2 and 4, creating a torque about the z-axis.

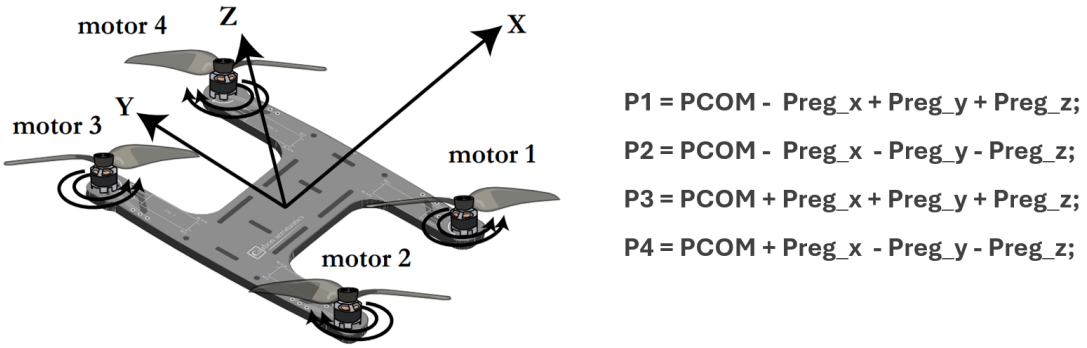


Figure 9: Quadcopter maneuvering strategy. All terms are measured in microseconds. The resulting motor power values are sent to the ESCs as PWM signals with specific pulse widths, where higher pulse widths correspond to higher RPM and greater lift. Image from [10]

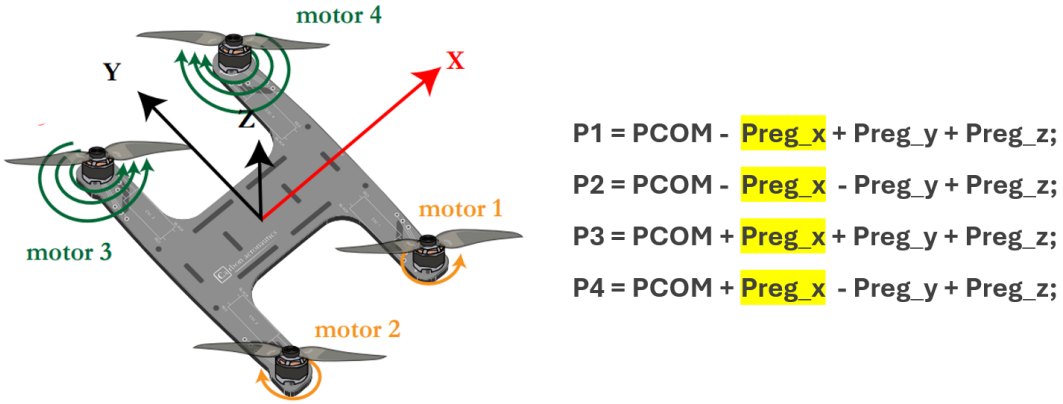


Figure 10: Achieving rotation around the horizontal x-axis (roll) by adjusting the P_{regx} term in the motor power equations. In this example, increasing the P_{regx} term causes motors 1 and 2 to slow down, while motors 3 and 4 speed up, resulting in the drone tilting accordingly. Pitch and yaw are achieved similarly by varying their respective terms. Image from [10].

3.4 Controller Implementation

Our specific controller is based on the simplified state space model in section 2.4.2, meaning it controls the angle and angular acceleration of the drone, rather than the speed. The controller is also a type of cascaded controller as describes in section 2.3. The complete multidimensional control loop is shown in figure 11, where $\vec{\alpha} = [\phi, \psi, \theta]$. Note that when this controller was implemented in code it was implemented as three single variable controllers running in parallel, one for roll, pitch and yaw. This was done to avoid matrix operations which greatly reduce the computational complexity. Both the rate controller and angle controller seen in figure 11 are PID controllers that have been tuned manually. First, the rate controller was tuned and tested, and then the angle controller was implemented. To get a better estimation of the angle for the angle controller a Kalman filter was used. This filter combines the measurements from an accelerometer and a gyroscope to obtain a better estimate of the angle than would have been possible using the sensors on their own. The downside to this is that the outer, angle control loop using the Kalman filter as its observer becomes rather slow, therefore, the inner control loop directly measuring and controlling angular acceleration is used for most of the stabilizing, while the outer slower loop makes sure the drone can approach any desired reference angles, a process which does not have to be as responsive. As can be read in section 1 this is a common way of using cascaded controllers.

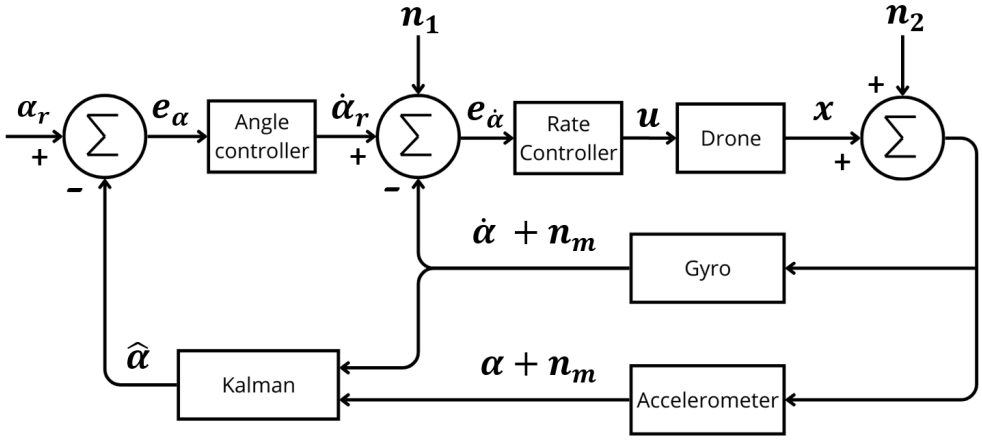


Figure 11: Block schedule of the complete multidimensional control loop. Note that $\vec{\alpha} = [\phi, \theta]$

Since the yaw is easier to control and not as crucial for stable flight it is controlled using a single rate based PID controller rather than the cascaded control structure in figure 9

Note: The complete code for the drone and the control loop can be found on GitHub [3]

3.5 Sensor Calibration

To calibrate our IMU sensor, we adopted the robust and easily implementable method described by Tedaldi et al. [9]. This method employs a multi-position scheme to estimate the scale factors, biases, and axis misalignments for both accelerometers and gyroscopes without requiring external equipment. By repositioning the IMU in various static orientations, the method leverages the stability of the gravity vector during static intervals to calibrate the accelerometers. The results from this calibration are then utilized to calibrate the gyroscopes through angular velocity integration.

For our implementation, we employed the MATLAB script provided in the GitHub repository *imu_tk_matlab* by JzHuai, [4] which applies this calibration method within a MATLAB framework. This script streamlined our calibration process and facilitated seamless integration with our dataset. We extend our gratitude to JzHuai for making this resource available.

3.6 Simplified Attitude Estimation

In order to estimate the attitude (pitch, yaw, and roll) of the quadcopter, a combination of an accelerometer and a gyroscope is typically employed. The accelerometer measures the angles of the axes by using gravity as a reference. This method provides accurate results when the quadcopter is stationary, as gravity is the dominant force acting on it. However, its performance deteriorates when the quadcopter is in motion due to the influence of external accelerations and vibrational noise from the motors.

The angle measurements are derived from the accelerometer's three axes, where a_x, a_y, and a_z represent the components of gravitational acceleration along with any external accelerations acting on the respective axes. These measurements are used to compute the accelerometer's inclination according to Equations 26 and 27.

[3] Project GitHub page: `https://github.com/ElektroJonas/DIY-Quadcopter/tree/main`

[4] The GitHub repository by JzHuai is available at: `https://github.com/JzHuai0108/imu_tk_matlab/tree/master`

$$\theta_k^{(\text{roll})} = \arctan\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \tag{26}$$

$$\phi_k^{(\text{pitch})} = -\arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \tag{27}$$

Another way to measure attitude is to use the gyroscope, which measures angular velocity. To estimate the angle, the angular velocity can be integrated over time. This method is unaffected by the drone's translational motion; however, integrating small measurement errors over time leads to a bias which worsens over time and causes the drone to drift.

To get a better estimation of the roll pitch and yaw a kalman filter, see section 2.5, can be implemented to utilise both the gyroscope and accelerometer. Since the Kalman filter needs to be implemented in code, the state space model first needs to be discretized. The simplfied state spaced model from section 2.4.2 can be discretized according to section 2.2.1 using the following approximations.

$$F = e^{A*T_s} = \sum_{k=0}^{\infty} \frac{1}{k!}(A * T_s)^k \approx I + A * Ts \tag{28}$$

$$G = (e^{ATs} - I)A^{-1}B \approx (I + A * Ts - I)A^{-1}B = Ts * B \tag{29}$$

The approximations in 28 and 29 work because the system is sampled at 250Hz meaning $Ts = 0.004 \implies A*Ts << 1 \implies \sum_{k=0}^{1} \frac{1}{k!}(A*T_s)^k >> \sum_{k=2}^{\infty} \frac{1}{k!}(A*T_s)^k$ The resulting state space model can be seen below.

$$\begin{bmatrix} \phi_k \\ \theta_k \\ \psi_k \\ \dot{\phi}_k \\ \dot{\theta}_k \\ \dot{\psi}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & T_s & 0 \\ 0 & 0 & 1 & 0 & 0 & T_s \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_{k-1} \\ \theta_{k-1} \\ \psi_{k-1} \\ \dot{\phi}_{k-1} \\ \dot{\theta}_{k-1} \\ \dot{\psi}_{k-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{T_s*d}{I_x} & 0 & -\frac{t_s*d}{I_x} \\ \frac{T_s*d}{I_y} & 0 & -\frac{T_s*d}{I_y} & 0 \\ -\frac{T_s*c}{I_z} & \frac{T_s*c}{I_z} & -\frac{T_s*c}{I_z} & T_s\frac{c}{I_z} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{30}$$

$$\begin{bmatrix}\phi_k\\ \theta_k\\ \psi_k\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\end{bmatrix} \begin{bmatrix}\phi_{k-1}\\ \theta_{k-1}\\ \psi_{k-1}\\ \dot{\phi}_{k-1}\\ \dot{\theta}_{k-1}\\ \dot{\psi}_{k-1}\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix} \begin{bmatrix}F_1\\ F_2\\ F_3\\ F_4\end{bmatrix} \tag{31}$$

Based on the simplified, discretized state-space model the roll pitch and yaw can be rewritten into first-order equations. This is beneficial because it enables the kalman filter to be broken down into multiple one dimensional filters avoiding the increased computational complexity that comes with matrix multiplications. For example, roll can be isolated like this:

$$\phi_k = \phi_{k-1} + Ts\dot{\phi}_{k-1} \tag{32}$$

$$\theta_k = \theta_{k-1} + Ts\dot{\theta}_{k-1} \tag{33}$$

$$\psi_k = \psi_{k-1} + Ts\dot{\psi}_{k-1} \tag{34}$$

With this set of fist order differential equations the kalman equations from section 2.5 can now be applied on pitch yaw and roll individually. Instead of using the system model to estimate the angular velocities we remove them from the state vector and measure them with the gyroscope.

$$\hat{\theta}_{k|k-1} = \hat{\theta}_{k-1|k-1} + T_s\hat{\dot{\theta}}_{k-1} \tag{35}$$

$$P_{k|k-1} = P_{k-1|k-1} + Q \tag{36}$$

$$\tilde{y}_k = y_k - \hat{\theta}_{k|k-1} \tag{37}$$

$$S_k = P_{k|k-1} + R \tag{38}$$

$$K_k = \frac{P_{k|k-1}}{S_k} \tag{39}$$

$$\hat{\theta}_{k|k} = \hat{\theta}_{k|k-1} + K_k\tilde{y}_k \tag{40}$$

$$P_{k|k} = P_{k|k-1}(1 - K_k) \tag{41}$$

Note: y is the estimation of the angle based on measurements from the accelerometer using equation 27 or 26. Were Q is the auto covariance process noise $E[\omega^2]$, and R is the auto covariance of the measurement noise $E[v^2]$. The noises are assumed to be zero mean and white.

Note: The initial value of P is assumed to be 0.

3.7 Extended Attitude Estimation

An improved Kalman filter can be achieved by incorporating the bias state ($\dot{\theta}_b$) in addition to the angle state (θ) derived in Section 3.6 [11] [12]. The primary objective is to enhance the accuracy of angle estimation by mitigating drift errors and compensating for biases inherent in gyroscope measurements.

A comprehensive explanation of the theoretical framework and mathematical formulation of the Kalman filter is provided in Section 2.5.

3.7.1 System and Measurement Models

As described in [12], the system state is represented as:

$$x = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}, \tag{42}$$

where θ is the angle, and $\dot{\theta}_b$ is the gyroscope bias. The inclusion of the bias in the state vector allows the filter to estimate and compensate for gyroscope drift over time. It should be noted that this state space representation only accounts for angle around one angle, so one angle and bias state is needed for roll, pitch and yaw.

The prediction step uses a state-space model:

$$x_k = Fx_{k-1} + B\dot{\theta}_k + w_k, \tag{43}$$

where the state transition matrix F and control-input model B are defined as:

$$F = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}. \tag{44}$$

The structure of F accounts for the fact that the angle θ is influenced by the gyroscope measurement ($\dot{\theta}$) over time, while the bias $\dot{\theta}_b$ remains constant unless updated. Specifically: The term $-\Delta t$ in F represents the integration of the gyroscope bias over time, which is subtracted from the measured angular velocity to estimate the unbiased angle. The control-input model B integrates the angular velocity ($\dot{\theta}_k$) over the time step Δt to update the angle.

The measurement model maps the system state to the accelerometer output:

$$y_k = Hx_k + v_k, \tag{45}$$

where:

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{46}$$

This structure reflects that the accelerometer provides a direct measurement of the angle θ based on its inclination but offers no direct information about the gyroscope bias.

Summarizing all the steps, the Kalman filter can be expressed as follows:

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k + w_k, \tag{17}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k + v_k. \tag{47}$$

3.7.2 Tuning of Covariance Matrices

The performance of the Kalman filter depends on the tuning of the process noise covariance (Q) and measurement noise covariance (R). The tuning process was based on the method used for tuning the covariances of a Kalman filter described by S. Y. Song et al. [11].

The gyroscope variance ($Q_{22} = \sigma_{\text{gyro}}^2$) was experimentally measured. The standard deviation (σ_{gyro}) was found to be:

$$\sigma_{\text{gyro}} = \sqrt{3}\,^\circ/\text{s}$$

The accelerometer variance ($R = \sigma_{\text{acc}}^2$) was experimentally measured. The standard deviation (σ_{acc}) was found to be:

$$\sigma_{\text{acc}} = \sqrt{12}\,^\circ$$

Q_{11}, representing noise in the angle estimation, was chosen experimentally. Following similar studies, $Q_{11} = \sigma^2_{\text{angle}}$ was set close to the gyroscope variance. The chosen standard deviation was:

$$\sigma_{\text{angle}} = \sqrt{3.5}^\circ$$

The covariance matrices were defined as:

$$Q_k = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{bmatrix} \Delta t, \quad R = \begin{bmatrix} \sigma^2_{\text{acc}} \end{bmatrix} \tag{48}$$

The presence of Δt (the time step) in the process noise covariance matrix Q reflects the influence of time on the system's uncertainty. Specifically, Q represents the uncertainty in the state prediction due to the process noise over the Δt time interval [5].

The initial state covariance (P_0) was initialized as:

$$P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{49}$$

This choice reflects the expectation that the system starts in a known state ($\theta = 0, \dot{\theta}_b = 0$) due to prior calibration or assumptions. Since no better initial guess was available, setting P_0 to zero was appropriate.

3.8 Altitude Estimation

Accurate height estimation is crucial for maintaining stable hover control in quadcopter drones. This section evaluates two algorithms designed for height estimation, both employing sensor fusion techniques to integrate measurements from an IMU and a barometer. These methods are tailored to mitigate sensor drift and noise, providing reliable altitude estimates essential for the drone's control system.

3.8.1 Two-Step Kalman Filter

The algorithm implemented [14], utilizes two Kalman filters in a cascaded manner. The initial Kalman filter estimates the attitude of the drone, providing roll and

[5] One might expect Δt to be squared as variance calculations involve squaring estimates. However, this is not the case, as explained by Thomas Jespersen in this thread `https://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/` (which also contains a good explanation of angle estimation using a Kalman filter) and in Appendix E of Joan Solà [13]

pitch angles necessary for compensating gravitational acceleration. The second Kalman filter estimates the vertical position and velocity by fusing barometric pressure data and IMU measurements. A ZUPT mechanism is incorporated in the second filter to constrain velocity drift during stationary phases. This method achieves robust altitude estimation, leveraging the Kalman filter's capability as an optimal observer.

It should be noted that the Kalman-Kalman approach is computationally intensive. Additionally, estimating the barometer's noise covariance matrix proved challenging. A experimental approach was employed for tuning of the Kalman filter, making it not a ideal Kalman filter, though it still provides good performance.

3.8.2 Zero Velocity Updates for Height Estimation

In scenarios where the drone hovers, ZUPT offers a robust solution for correcting drift errors in height estimation. The concept, originally developed for pedestrian tracking [15], applies zero velocity constraints during stationary phases to recalibrate velocity and mitigate position drift. This approach is well-suited for drones in hover mode, where acceleration in the vertical direction aligns closely with the Earth's normal vector and remains near zero.

Magnitude detection can be employed to identify stationary phases, as discussed in [16] and the theory section of this report 2.7.1. Although zero acceleration does not always imply zero velocity, this method effectively constrains velocity drift when the drone is in stable hover. Incorporating ZUPT into the height estimation filter provides retroactive corrections to velocity and position errors by leveraging correlations in the Kalman filter's covariance matrix [7]. This technique has been shown to enhance height estimation significantly in studies [14] [16].

3.8.3 System and Measurement Models

This filter operates as part of a cascaded Kalman Filter architecture. A preceding Kalman filter, described in section 3.6 and 3.7, estimates the drone's attitude, including roll (γ) and pitch (β), which are essential for compensating gravitational acceleration. These estimated angles are used to align the sensor frame with the navigation frame, ensuring accurate height and velocity predictions [14].

[14]Vertical height (h_k) and velocity (v_k) are represented by the following state vector:

$$\hat{x}_{k|k-1} = \begin{bmatrix} h_k \\ v_k \end{bmatrix} \tag{50}$$

The prediction equation is:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Gu_{k-1} + w_{k-1}, \tag{51}$$

where the state transition matrix A and control-input model B are defined as:

$$F = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}, \tag{52}$$

The input u_{k-1} represents the gravity-compensated vertical acceleration in the navigation frame and is calculated as:

$$u_{k-1} = [0\ 0\ 1] \cdot Ra_{k-1}, \quad R = R_z(\psi)R_y(\theta)R_x(\phi), \tag{53}$$

where $a_{k-1} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ is the acceleration vector measured in the sensor frame. The rotation matrix R transforms the sensor frame to the navigation frame, utilizing Kalman-estimated yaw (ψ), pitch (θ), and roll (ϕ). The rotation matrices are defined as:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \tag{54}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{55}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{56}$$

Using the rotation matrix R, the vertical acceleration in the navigation frame, u_{k-1}, simplifies to:

$$u_{k-1} = -\sin\theta \cdot a_x + \cos\theta\sin\phi \cdot a_y \cos\theta\cos\phi \cdot a_z. \tag{57}$$

Here, $a_{k-1} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ contains the accelerometer readings in the sensor frame. The calculated u_{k-1} represents the acceleration along the global z-axis (altitude direction) in the navigation frame.

The measurement model updates the predicted state using barometric pressure data. The measurement equation is:

$$y_k = H\hat{x}_{k|k-1} + v_k, \quad H = [1\ 0]. \tag{58}$$

Here, z_k represents the measured relative height, and v_k accounts for measurement noise. The barometric height h_{baro} is computed as:

$$h_{\text{baro}} = 44330 \cdot \left(1 - \left(\frac{P}{P_0}\right)^{0.19}\right) - h_{\text{init}}, \tag{59}$$

where P is the current pressure, P_0 is either the average atmospheric pressure or a calibrated pressure from a weather station, and h_{init} is the initial height. Due to the dependency of barometric pressure on weather conditions, altitude estimates exhibit a random walk rather than white noise, necessitating careful modelling of measurement noise to achieve optimal performance.

Summarizing all the steps, the Kalman filter can be expressed as follows:

$$\hat{x}_{k|k-1} = \begin{bmatrix} h_k \\ v_k \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h_k \\ v_k \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \cdot u_{k-1} + w_{k-1}, \tag{60}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} h_k \\ v_k \end{bmatrix}_k + v_k. \tag{61}$$

where u_{k-1} is the gravity-compensated vertical acceleration in the navigation frame, defined as:

$$u_{k-1} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ \sin\beta\sin\gamma & \cos\beta & \sin\beta\cos\gamma \\ \cos\beta\sin\gamma & -\sin\beta & \cos\beta\cos\gamma \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{62}$$

To address drift in velocity and position, the filter incorporates a Zero Velocity Update (ZUPT) mechanism. During stationary phases, identified by monitoring the norm of acceleration, the vertical velocity state is set to zero. This recalibration significantly mitigates drift errors and improves long-term stability.

3.8.4 Tuning of Covariance Matrices

The performance of the Kalman filter for altitude estimation depends on the tuning of the process noise covariance (Q) and measurement noise covariance (R). The altitude variance ($Q_{11} = \sigma^2_{\text{alt}}$), velocity variance ($Q_{22} = \sigma^2_{\text{vel}}$) and barometric variance ($R = \sigma^2_{\text{baro}}$) were tuned experimentally to achieve a good performance.

The variances used were as follows:

$$\sigma_{\text{alt}} = \sqrt{110.0}\,\text{m}, \quad \sigma_{\text{vel}} = \sqrt{110.0}\,\text{m/s}, \quad \sigma_{\text{baro}} = \sqrt{1300.0}\,\text{m}.$$

The covariance matrices were defined as:

$$Q_k = G \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{bmatrix} G^\top, \quad R = \begin{bmatrix} \sigma_{\text{baro}}^2 \end{bmatrix}. \tag{63}$$

The initial state covariance (P_0) was initialized as:

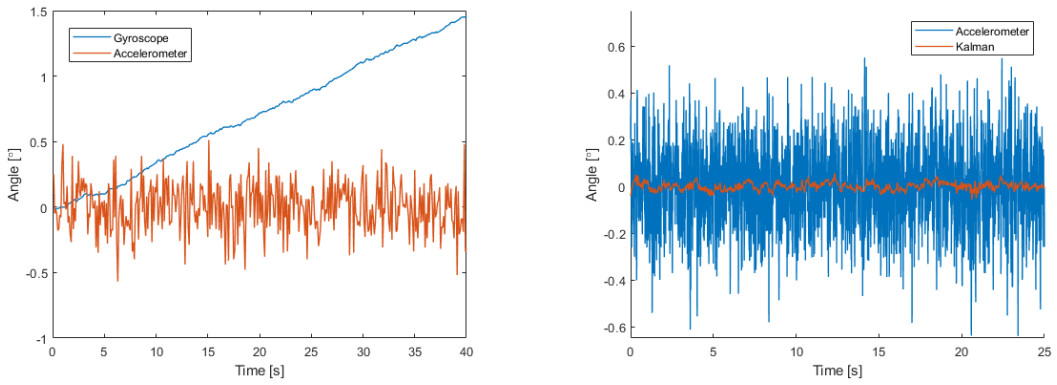
$$P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{64}$$

This initialization reflects an assumption of no prior knowledge about the altitude or velocity at startup. Since no better initial guess was available, this choice was appropriate for the system.

4 Results

4.1 Attitude Measurement

The Kalman filter effectively eliminates drift from the gyroscope integration while providing smoother and more reliable angle estimates with significantly reduced noise compared to accelerometer-based measurements, as illustrated in Figure 12.



(a) Angle accelerometer and gyroscope

(b) Angle accelerometer Kalman filter

Figure 12: Comparison of angle measurements before and after Kalman filtering in a stationary environment.

The filter demonstrates reliable tracking of motion, as presented in Figure 13, by significantly reducing noise without introducing substantial delay to the angle estimates. The noise level is notably lower than that observed in accelerometer-based inclination measurements. Furthermore, the Kalman filter effectively retains the low-noise characteristics of the gyroscope's integrated measurements while compensating for bias, which is particularly evident at the start and end of the dataset.

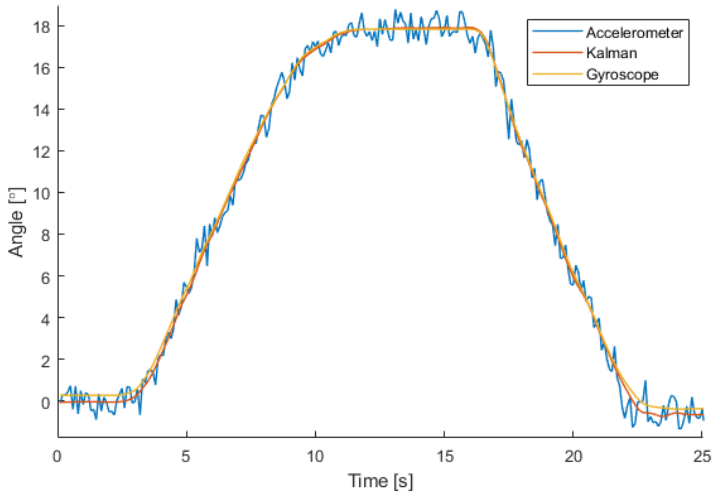


Figure 13: Performance during manual tilting motion, comparing accelerometer inclination, Kalman-filtered angles, and gyroscope integration.

4.2 Altitude Measurements

The altitude measurements were conducted with the drone's motors turned off, and the drone was manually moved up and down by approximately 30 cm. The results are presented in two figures, highlighting the performance of different measurement methods.

Figure 14 compares the altitude measurements from three methods: the standard Kalman filter, the Kalman filter enhanced with the Zero Velocity Update algorithm, and the barometer. The barometric measurements exhibit noticeable noise and a slight random walk, as expected from such sensors. The Kalman filter measurements show reduced noise but are prone to drift over time. By incorporating the ZUPT algorithm, the Kalman filter follows the expected up-and-down motion fairly well.

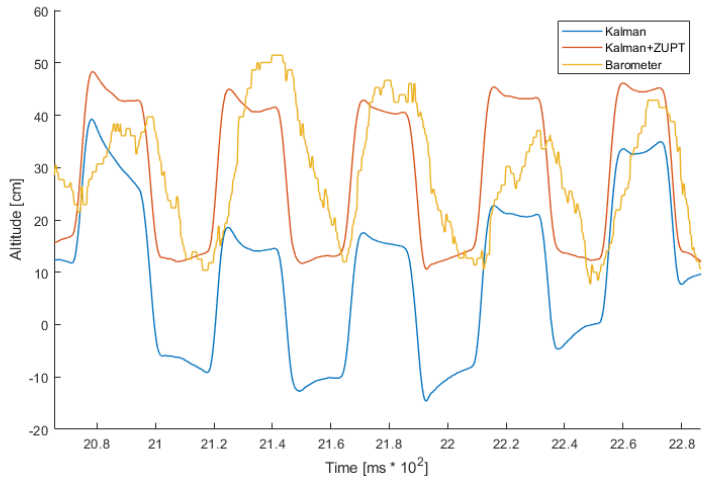


Figure 14: Comparison of altitude measurements: standard Kalman filter, Kalman filter with ZUPT, and barometer.

Figure 15 illustrates the comparison of the velocity state estimated by the Kalman filter, both with and without the ZUPT algorithm. When ZUPT is not utilized, the velocity estimate exhibits drift over time, indicating accumulating errors. In contrast, with ZUPT integrated, the velocity state remains stable and accurately represents the motion, effectively mitigating drift.

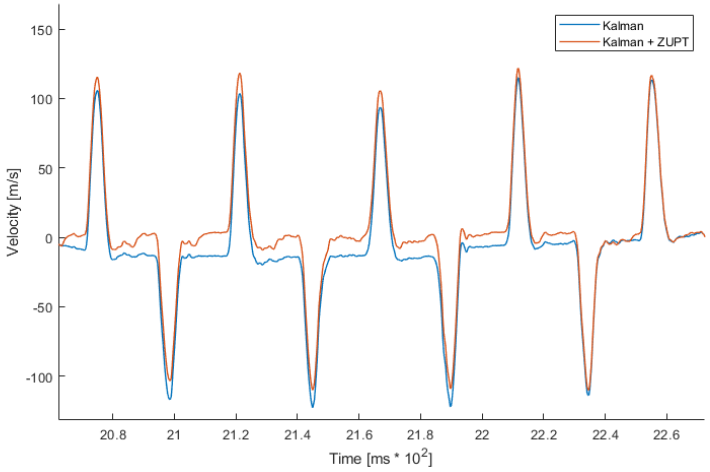


Figure 15: Comparison of velocity state estimation: Kalman filter with and without ZUPT algorithm.

4.3 Flight Performance

The quadcopter is capable of hovering and maneuvering in multiple directions, including up, down, forward, backward, left, right, as well as rotating around its z-axis. The quadcopter responds to remote commands, though its responsiveness is somewhat limited. More aggressive parameters were tested in the hope of getting a more responsive quadcopter, this did however introduce too much instability.

What is currently hardest to control the the altitude of the quadcopter. As mentioned in 3.8, an advanced height control system was initially attempted, but due to time constraints, it could not be fully implemented. As an alternative, the throttle input from the joystick (PCOM) was mapped to a narrower range of 40–60% of the maximum motor power. This adjustment was made to enhance sensitivity in the range where the quadcopter typically operates, as it lifts off the ground at approximately 45% throttle given its current weight. By focusing the input within this critical range, the pilot gains finer control over the quadcopter's vertical motion, particularly during low-altitude maneuvers, which is the case for our indoor quadcopter. However, this tradeoff reduces the available range for higher throttle values, limiting the maximum vertical acceleration.

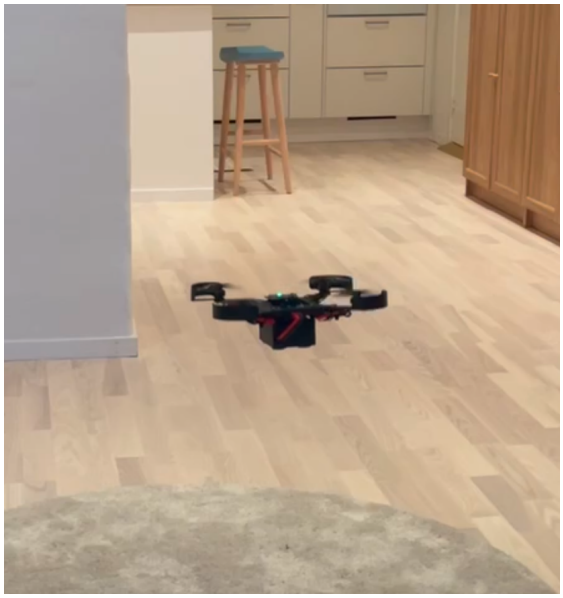


Figure 16: Demonstration of quadcopter flying. Video available here `https://www.youtube.com/watch?v=qHReMhCUwgA`

5 Discussion

5.1 Reflections and Lessons Learned

This project highlighted the trade-offs between complexity, time constraints, and scope. Decisions such as using off-the-shelf ESCs and development boards were necessary compromises that balanced feasibility with performance. The successful implementation of the custom remote and the cascaded control system underscores the importance of focusing on achievable milestones while maintaining flexibility in project planning.

Future iterations should explore integrating advanced components, such as ESCs with Oneshot protocol and an onboard BMS. Additionally, implementing a height control system and improving system modeling would further enhance the quadcopter's performance and reliability.

5.2 Project Specifications

5.2.1 Goal 1: Battery and Power Management

Comparing the first project goal outlined in Section 1.3 with the hardware described in Section 3.2.1 reveals that the quadcopter includes a battery pack but lacks a BMS. This decision was necessitated by the custom cylindrical battery packs failing to meet the required power density specifications, leading to the use of an off-the-shelf LiPo battery. The chosen LiPo battery is charged using a dedicated charger that manages most of the BMS functionalities, including balancing the individual cells.

Under-voltage protection was implemented in the drone to safeguard the battery pack; however, individual cell voltages are not monitored. This omission is mitigated by the charger's reliable balancing capabilities during each charging cycle. Although this solution addresses safety and performance needs effectively, integrating a BMS in future iterations would improve autonomy and operational safety, especially for prolonged flights.

5.2.2 Goals 2 and 3: Custom Electronics and PCBs

The second and third project goals—designing all electronics from scratch and integrating them onto custom-made PCBs—were only partially fulfilled. As shown in Figure 3, the drone's electronics consist of an experimental PCB, a microcontroller development board, and an MPU6050 development board. Initial plans to avoid development boards and consolidate all components onto a single custom PCB were abandoned due to time constraints.

The complexity of designing electronic speed controllers exceeded the project's scope, and off-the-shelf ESCs were used instead. However, the remote control, depicted in Figure 5, was successfully implemented on a custom-made PCB. This achievement enhanced the remote's reliability and professional appearance, demonstrating the potential for custom PCB design in future projects.

5.2.3 Goal 4: Advanced Flight Controller

We believe the fourth goal—implementing an advanced flight controller capable of reliable and sustained flight—was achieved. The cascaded controller described in Section 3.9, combined with the Kalman filter for attitude estimation, constitutes a reasonably advanced control system. While "advanced" is subjective, the use of sensor fusion and cascaded PID controllers aligns with the project's ambition to implement sophisticated control strategies.

However, the absence of a height controller, as mentioned in Section 4.6, remains a limitation. This omission affected the quadcopter's ability to maintain a consistent altitude autonomously. Instead, a narrower throttle range was used to improve pilot control over altitude during indoor flights. Addressing this gap in future designs would further enhance the system's functionality and usability.

5.2.4 Goal 5: Reliable and Sustained Flight

As discussed in Section 4.6, the quadcopter successfully demonstrated reliable and sustained flight with basic maneuvering capabilities. It was able to hover and move in multiple directions, including up, down, forward, backward, left, and right, as well as rotate around its z-axis. However, the drone's responsiveness was somewhat limited due to conservative control parameter tuning. While more aggressive parameters were tested, they introduced instability, highlighting the need for further optimization of control strategies.

6 Further Work

6.1 Implementation of Height Controller

Due to time constraints, the implementation of the height controller was not completed within the project timeline. Despite our efforts, we were unable to develop a fully functional version before the project's conclusion.

Future work on this project should focus on further development, testing, and tuning of the height controller to enable a reliable hover mode. This would build on the altitude estimation algorithm that was successfully implemented and detailed in this report. A well-tuned height controller is essential for achieving stable and precise altitude control, which is a critical component of the drone's overall performance.

6.2 Choice of ESC

In a future iterations of the drone, we would plan to explore using ESCs that support the Oneshot protocol. PWM operated ESC typically runs at a maximum control input frequency of 400 Hz, which limits the potential responsiveness of the control loop. Oneshot's higher update rate and reduced latency would better align with our control loop, allowing faster and more precise motor responses. This improvement could significantly enhance flight stability and agility.

6.3 Attitude Estimation

To improve altitude estimation, the algorithm in [17] can be implemented. This Kalman filter-based method estimates attitude and external accelerations using inertial sensors, incorporating an external acceleration model to reduce errors in dynamic conditions. By employing a first-order low-pass filtered noise process, it ensures accurate gravitational component estimation even during rapid motion, enhancing performance in both static and dynamic scenarios.

6.4 Altitude Estimation

An alternative approach [16], *A Two-step Kalman/Complementary Filter for Estimation of Vertical Position Using an IMU-Barometer System*, addresses the drawbacks of the two-step Kalman-Kalman filter [14] by replacing the second Kalman filter with a computationally efficient complementary filter. The attitude estimation remains the same, relying on the Kalman filter, while the complementary filter fuses vertical acceleration and barometric pressure data to estimate altitude.

Furthermore, this method integrates ZUPT for improved accuracy, allowing the system to reset velocity during stationary phases and effectively reduce drift errors.

The complementary filter significantly reduces computational complexity, requiring only 12.5

6.5 System Modeling

Much of control theory relies on model-based approaches to fully leverage advanced control techniques. Methods such as root locus, pole placement, and optimal control frameworks like LQG, H_2, and H_∞ require an accurate system model for effective implementation. In this project, identifying the system model for the drone was deemed too time-consuming given the project's scope. Additionally, it was anticipated that a quickly derived model would likely introduce significant model errors (i.e., discrepancies between the estimated system model and the true dynamics), limiting the potential improvements offered by these advanced model-based approaches. Consequently, a simpler experimental approach using a PID controller was selected, as it aligned better with the project's goals and scope.

That said, there exists a wealth of research and theory on quadcopter system modeling, which could be explored in future work. Developing an accurate model of the drone would enable the application of more advanced control techniques and potentially yield significant performance improvements. Some interesting papers that could serve as a foundation for future work on system modeling and identification include Szabó (2019) [18], Sa and Corke (2012) [19], Mustapa et al. (2014) [20], and Wei et al. (2015) [21].

6.6 Noise Reduction

A significant challenge in the application of an IMU on a quadcopter is its operation in a highly noisy environment. Vibrations from the motors, combined with inherent sensor noise and disturbances from wind and turbulence during flight, contribute to substantial noise in the measurements. This project employs digital filtering low-pass filters integrated within the IMU's internal digital processor, as the primary method for noise reduction.

For our setup, using a 4800 KV motor and a 3-cell voltage setup, the maximum motor RPM is calculated as:

$$\text{Motor RPM} = 4800 \text{ RPM/V} \times 11.1 \text{ V} = 53,280 \text{ RPM},$$

which corresponds to a frequency of:

$$\text{Motor Frequency} = \frac{53,280}{60} \approx 888 \text{ Hz},$$

along with its harmonics. According to the Nyquist theorem, the sampling frequency should be at least twice this value (1776 Hz) to avoid aliasing. However, the MPU6050 sensor used in this project has a maximum sampling frequency of 1000 Hz, which is insufficient to meet this criterion. Consequently, aliasing effects distort the measurements.

Two potential solutions were considered but not implemented due to time constraints:

1. **Lower Motor RPM:** Increasing the propeller size and using a lower KV motor would reduce the motor RPM, thereby decreasing noise and its harmonics, as sugested by T. Brungart et al. [22].

2. **Anti-Aliasing Filters:** Implementing an anti-aliasing filter attenuates high-frequency noise before sampling. This can be achieved using mechanical low-pass filters, such as vibration dampers, whose design and analysis are discussed in the paper by C. Yilmaz et al. [23]. The open-source ArduPilot project provides examples of vibration damping solutions that demonstrate practical implementations of such filters[6].

7 Code Optimization and Performance Improvements

The quadcopter code could be further optimized to execute the control loop at a higher frequency, potentially enhancing both responsiveness and stability. One approach to achieve this is by translating the Arduino code into pure C. Initially, this was considered; however, as the computational time for a single iteration of the loop in Arduino code is only 2 ms, and the ESCs can only be updated every 4 ms, such optimization was deemed unnecessary. Furthermore, running the Kalman filter at a higher frequency than the rest of the control loop was briefly explored, but no significant performance improvements were observed.

Additional optimizations can be achieved by replacing the Kalman filter with a complementary filter, as discussed in Section 6.4. The switch from a Kalman filter to a complementary filter or a Madgwick filter [11] could also be considered for attitude estimation, significantly reducing computational load.

[6]Vibration damping solutions by the open-source ArduPilot project: `https://ardupilot.org/copter/docs/common-vibration-damping.html`

Performance improvements can also be realized by using optimized libraries, such as the Fast Trig library [7], which leverages lookup tables for trigonometric calculations. This approach provides a notable boost in performance at the expense of a slight increase in calculation error.

[7]Fast Trig library GitHub repository: `https://github.com/RobTillaart/FastTrig`

References

[1] T. Glad and L. Ljung, *Reglerteori Flervariabla och olinjära metoder*, 2nd ed. Lund, Sweden: Studentlitteratur AB, 1997.

[2] A. Vasičkaninová, M. Bakošová, and J. Oravec, "Chapter 13 - fuzzy control of heat exchangers in series using complex control structures," in *Advanced Analytic and Control Techniques for Thermal Systems with Heat Exchangers*, L. Pekař, Ed. Academic Press, 2020, pp. 287–306. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012819422500013X

[3] J. Ren, D.-X. Liu, K. Li, J. Liu, Y. Feng, and X. Lin, "Cascade pid controller for quadrotor," in *2016 IEEE International Conference on Information and Automation (ICIA)*, 2016, pp. 120–124.

[4] Z. Tahir, M. Jamil, S. Liaqat, L. Mubarak, W. Tahir, and S. Gilani, "State space system modeling of a quad copter uav," *Indian Journal of Science and Technology*, vol. 9, 07 2016.

[5] J. Björsell and A. Özçelikkale, "Signal processing 1te651: Kalman filtering," 2024, course material for Signal Processing 1TE651.

[6] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, ser. Prentice-Hall Signal Processing Series. Upper Saddle River, New Jersey: Prentice Hall PTR, 1993, chapter 13: Kalman Filters.

[7] J. Wahlström and I. Skog, "Fifteen years of progress at zero velocity: A review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1139–1151, 2021.

[8] I. Skog, P. Handel, J.-O. Nilsson, and J. Rantakokko, "Zero-velocity detection—an algorithm evaluation," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 11, pp. 2657–2666, 2010.

[9] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3042–3049.

[10] *Carbon Aeronautics Quadcopter Manual*, Online Guide, 2023, licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License. [Online]. Available: https://github.com/CarbonAeronautics

[11] S. Y. Song, Y. Pei, and E. T. Hsiao-Wecksler, "Estimating relative angles using two inertial measurement units without magnetometers," *IEEE Sensors Journal*, vol. 22, no. 20, pp. 19 688–19 699, 2022.

[12] P. Gui, L. Tang, and S. Mukhopadhyay, "Mems based imu for tilting measurement: Comparison of complementary and kalman filter based data fusion," in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, 2015, pp. 2004–2009.

[13] J. Solà, "Quaternion kinematics for the error-state kalman filter," 2017. [Online]. Available: https://arxiv.org/abs/1711.02508

[14] S. Zihajehzadeh, T. J. Lee, J. K. Lee, R. Hoskinson, and E. J. Park, "Integration of mems inertial and pressure sensors for vertical trajectory determination," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 3, pp. 804–814, 2015.

[15] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38–46, 2005.

[16] J. K. Lee, "A two-step kalman/complementary filter for estimation of vertical position using an imu-barometer system," *Journal of Sensor Science and Technology*, vol. 25, no. 3, pp. 202–207, 05 2016.

[17] J. K. Lee, E. J. Park, and S. N. Robinovitch, "Estimation of attitude and external acceleration using inertial sensor measurement during various dynamic conditions," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 8, pp. 2262–2273, 2012.

[18] A. P. Szabo, "System identification and model-based control of quadcopter uavs," Master's thesis, Wright State University, Dayton, OH, USA, 2019.

[19] I. Sa and P. Corke, "System identification, estimation and control for a cost effective open-source quadcopter," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2202–2209.

[20] Z. Mustapa, S. Saat, S. H. Husin, and T. Zaid, "Quadcopter physical parameter identification and altitude system analysis," in *2014 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*, 2014, pp. 130–135.

[21] W. Wei, M. Tischler, and K. Cohen, "System identification and controller optimization of a quadrotor uav," in *Annual Forum Proceedings - AHS International*, vol. 4, 05 2015.

[22] T. Brungart, S. Olson, B. Kline, and Z. Yoas, "The reduction of quadcopter propeller noise," *Noise Control Engineering Journal*, vol. 67, p. 252, 08 2019.

[23] C. Yilmaz and N. Kikuchi, "Analysis and design of passive low-pass filter-type vibration isolators considering stiffness and mass limitations," *Journal of*

Sound and Vibration, vol. 293, no. 1, pp. 171–195, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022460X05006632

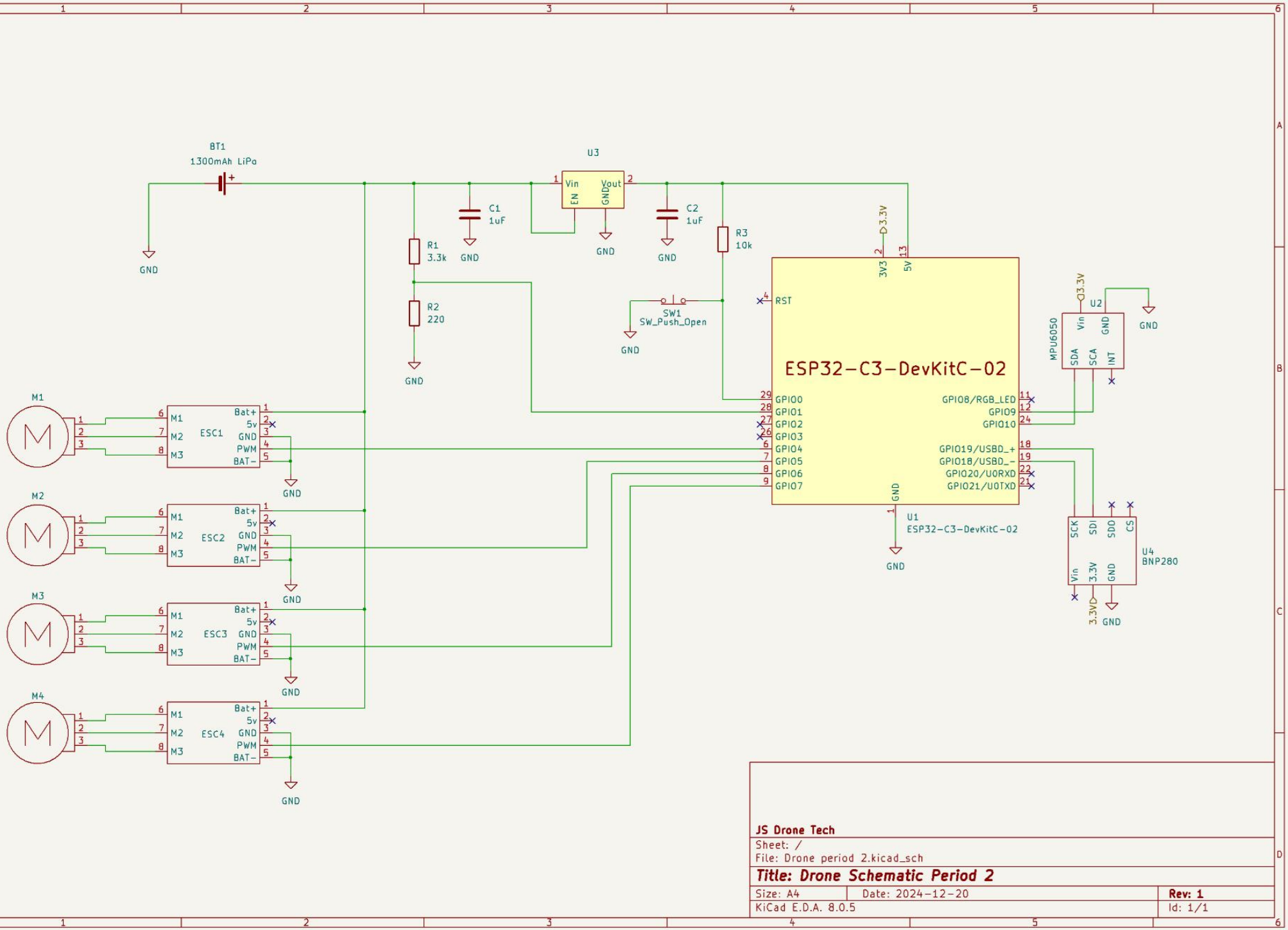
Appendix A: Drone Code

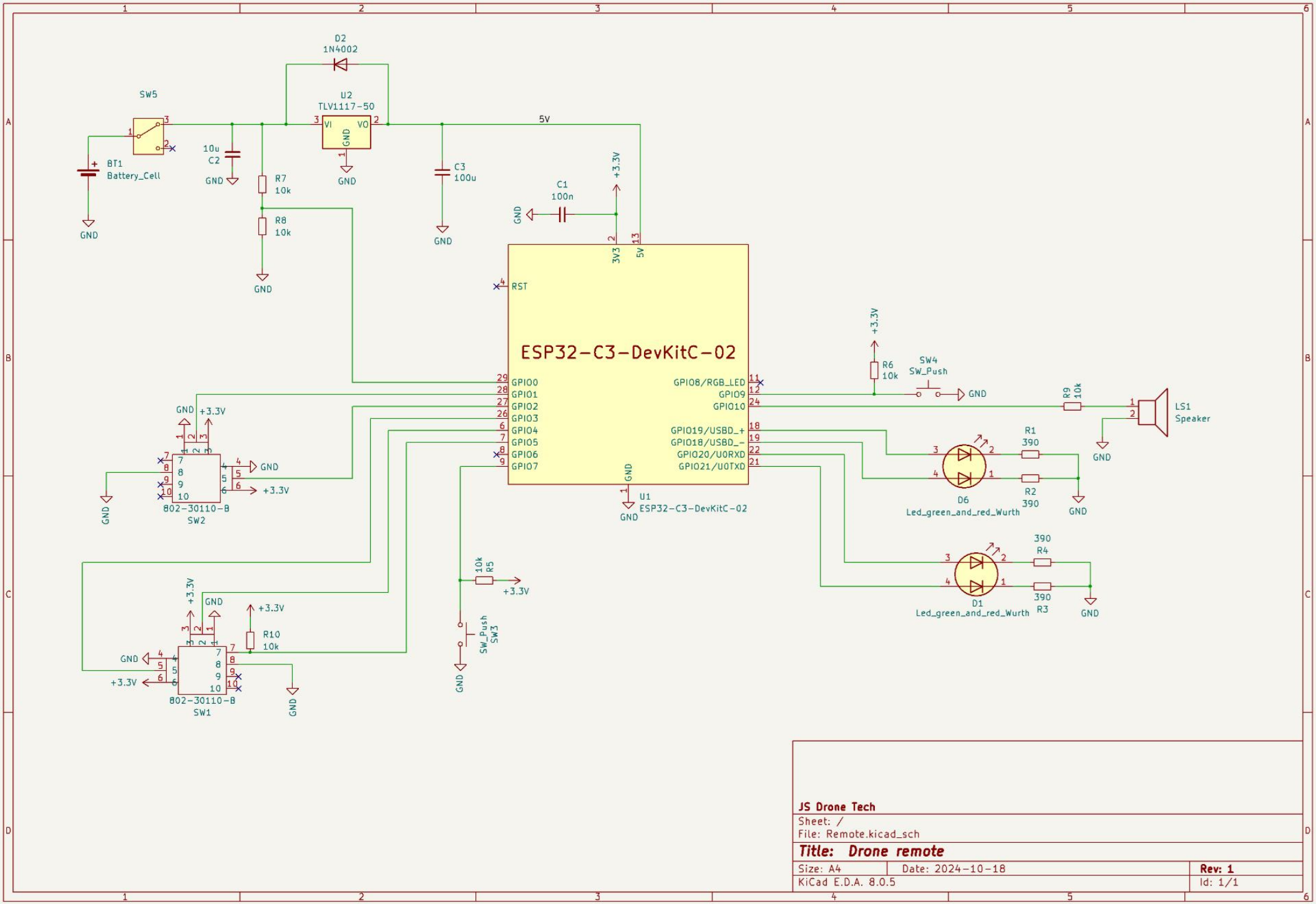
The complete source code for the drone control system, including the controller, is available in the following GitHub repository:

`https://github.com/ElektroJonas/DIY-Quadcopter/tree/main`

Appendix B: Schematics

The following page contains the electrical schematics for the drone. The schematics are presented in two sections: first, the drone's electronics, followed by the remote control system:


BT1
1300mAh LiPo
GND
U3
Vin
Vout
EN
GND
C1
1uF
C2
1uF
R1
3.3k
R2
220
R3
10k
SW1
SW_Push_Open
3.3V
5V
3V3
RST
ESP32-C3-DevKitC-02
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8/RGB_LED
GPIO9
GPIO10
GPIO19/USBD_+
GPIO18/USBD_-
GPIO20/U0RXD
GPIO21/U0TXD
U1
ESP32-C3-DevKitC-02
U2
MPU6050
SDA
SCA
INT
U4
BNP280
SCK
SDI
SDO
CS
M1
M2
M3
M4
ESC1
ESC2
ESC3
ESC4
Bat+
5v
PWM
BAT-
JS Drone Tech
Sheet: /
File: Drone period 2.kicad_sch
Title: Drone Schematic Period 2
Size: A4
Date: 2024-12-20
Rev: 1
KiCad E.D.A. 8.0.5
Id: 1/1



JS Drone Tech		
Sheet: /		
File: Remote.kicad_sch		
Title: Drone remote		
Size: A4	Date: 2024-10-18	**Rev: 1**
KiCad E.D.A. 8.0.5		Id: 1/1