Project Phase Report 2

Open Advanced Course in Embedded Systems

Jonas Gartner
Sigge Axelsson

November 8, 2024

1 Summary

Over the past two weeks we have started working on a controller, learned how to program esp32 and how to make them communicate with each other and we have learned how the practical implement of digital PID controllers is done.

2 Background

A drone is an unmanned aerial vehicle (UAV) that can be remotely controlled or fly autonomously using pre-programmed flight plans. It typically consists of a frame, motors, propellers, a battery, and sensors like GPS and cameras. Drones are used for various purposes, including photography, delivery, surveillance, and scientific research. They range from small consumer models to large military-grade systems.

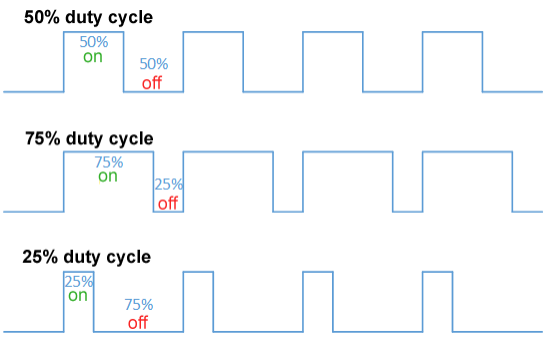
3 Theory

3.1 Pulse Width Modulation (PWM)

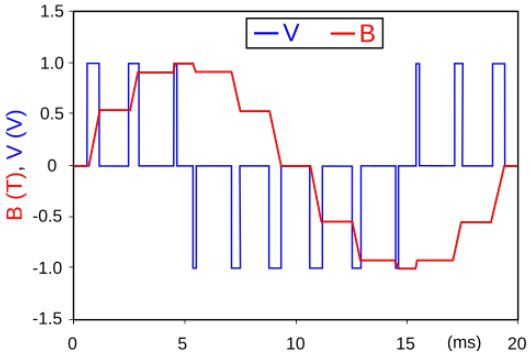
Pulse width modulation (PWM) is a technique used to control the amount of power delivered to a load by varying the width of the pulses in a periodic waveform. The key parameter in PWM is the duty cycle, described in equation 1, which refers to the proportion of the time during which the signal is "on" versus "off" within each cycle, seen in figure 1a. A higher duty cycle means the signal stays on longer during each period, delivering more power, while a lower duty cycle reduces the power, which can be seen in the resulting voltage in figure 1b. PWM is widely used in applications such as motor control, signal generation, and power regulation due to its efficiency and flexibility.

The duty cycle (D) is given by equation 1 where t_{on} is the time the signal is high (on) during one cycle, and T is the total period of the cycle.

$$D = \frac{t_{on}}{T} \tag{1}$$



(a) Examples of different duty cycles: $D = 50\%$, $D = 75\%$, and $D = 25\%$.



(b) PWM in an ideal inductor: Voltage pulses result in a sine-like current.

Figure 1: Illustration of pulse width modulation (PWM) with varying duty cycles and its application in an ideal inductor. Images sourced from Wikipedia.

3.2 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 2 describes a general PID controller in continuous time, and equation 3 describes a PID controller in discrete time:

$$Input_{\text{motor}}(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{2}$$

$$Input_{\text{motor}}(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i)\Delta t + K_d \frac{e(k) - e(k-1)}{\Delta t} \tag{3}$$

4 New components

4.1 Electronic Speed Controller

The Turnigy Plush-32 12A (2 4S) Brushless Speed Controller, shown in Figure 2, operates through PWM. The standard and recommended PWM frequency for the ESC is $f = 50hz$ resulting in a period $T = 200000us$, were a maximum throttle is achieved at a duty cycle of $D = 10\%$ resulting in a $T_{on} = 2000\mu s$ and the lowest is at the duty cycle $D = 5\%$ $T_{on} = 1000\mu s$.

Before using the electronic speed controller (ESC) you need to calibrate it, set the duty cycles for max and lowest throttle. This is done by sending the PWM signal coresponfing to the maximum thottle to the ESC and the powering on the ESC, this will set the maximum throttle indicated by a beeping sound. After the sound indication you send the PWM signal corresponding to low throttle speed, this is hten indicated by a beeping sound followed by a sound indicating that the ESC has been calibrated.

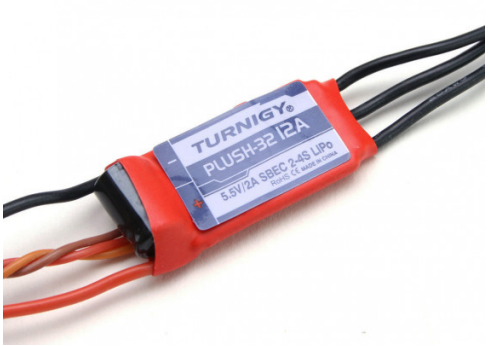


Figure 2: Turnigy Plush-32 12A (2 4S) Brushless Speed Controller

4.2 ESP32

ESP32 is a series of 32 bit microcontrollers with integrated bluetooth and wifi. Compared with the commonly used ATMEGA328P, which used in most arduinos, the ESP32 has more bits it can make the same computations in fewer clock cylces. Also, the ESP32 can run at 240Mhz while the ATMEGA is limited to 20MHZ which also contributes to it being faster.

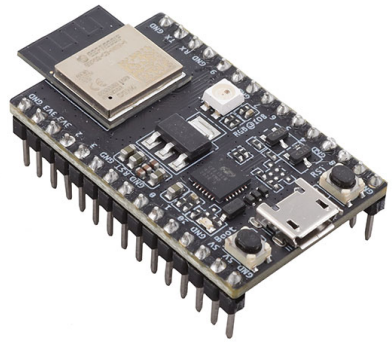


Figure 3: ESP32 micro controller, model: ESP32-C3-DEVKITM-1

5 Designing digital control systems

To find control parameters for a controller a model of the system is required. This model can be derives analytically from physics or it can be found using methods for system identification. However, it can often be possible to find good control parameters just by trail and error which is therefore often the go to method for many engineers. Another thing to take into consideration is the effects of implementing the controller on a computer, thus making the system discrete. There is a lot of theory about how to design controllers for digital systems. However, the easiest approach is just sampling the system at really high frequencies. The drawback of this method is that it requires a faster processor than a well designed digital controller.

5.1 Simple model of dynamics and controller

The current model we use to control the drone is shown in figure 4. The power to the motors, P_{M1} and P_{M2}, consists of a common term, P_{common}, which keeps the drone hovering. This term can be increased or decreased to make the drone ascend or descend. Additionally, there is a second term, P_{reg}, which is added to one motor and subtracted from the other. This causes the drone to rotate around its horizontal axis.

Instead of controlling the speed directly, we plan to control the tilt angle of the drone. As a side effect, we expect the drone to start moving in the direction of its tilt, similar to how a car driver controls the fuel supply rather than the speed directly. The driver becomes part of the control system, as their visual feedback is necessary to adjust the fuel supply and maintain a constant speed. The same principle will apply to our drone.

We have encountered several challenges while implementing this. The first issue is the difficulty in accurately measuring the angle. We have been using an accelerometer that measures acceleration in the x, y, and z directions, and combining this data with basic trigonometry (see figure 5) to calculate the drone's horizontal angle relative to gravity. This method works because we always know the direction and magnitude of gravity, and we can measure its components along the x, y, and z axes. From these components, we can infer the angle. However, this approach only works when the drone is stationary. Any acceleration other than gravity affects the drone, causing our calculations to break down.

To solve this problem, we have started developing a controller that operates based on the drone's angular acceleration rather than its angle. The effect will be the same as described earlier: the drone will maintain a specified angle. Although we won't directly know the angle, it doesn't matter, as what we are truly concerned with is the drone's speed.

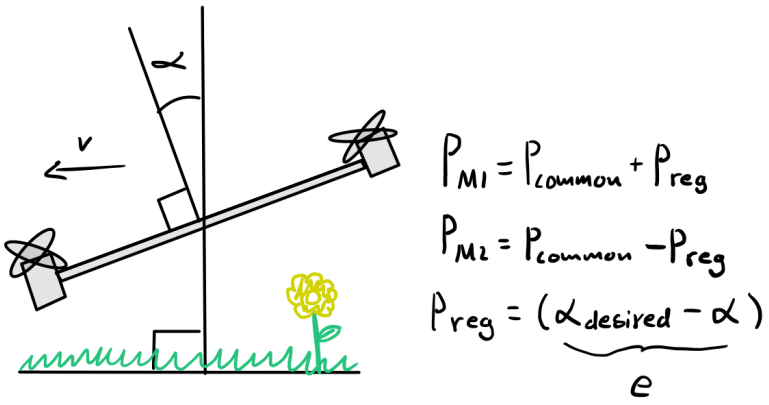


Figure 4: First draft of a model used to control speed of drone

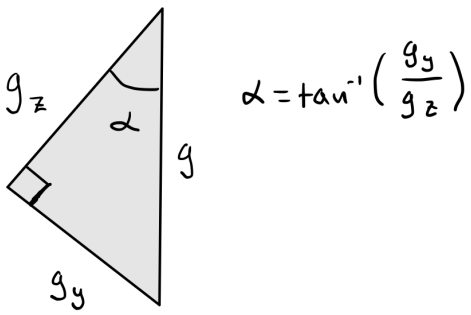


Figure 5: Model used to calculate anlge based on gravity. This works because gravity always point straight towards the ground.