Introduction of a Flight Controller into a Swashplateless Control System for Small Helicopters

Mason McMahon   
*Dept. of Computer Science and Engineering  
Wright State University*Fairborn, OH  
mcmahon.mason22@gmail.com

*Abstract*— This paper explores the theory, design, and implementation of a control scheme for altering blade pitch without the use of a mechanical swashplate. By modulating the duty cycle sent into the motor we were able to achieve cyclical control. This allows for full pitch, roll, and thrust control without the need for any other actuators to be present in the system.

# Motivation & Problem Statement

A helicopter is a type of rotor craft capable of generating lift and thrust through horizontal spinning rotors. In order to achieve full 3-dimensional locomotion, a control scheme must be used to vary the pitch of the rotor. Traditionally, large helicopters will use a swashplate, a mechanical device that transmits control inputs from the stationary fuselage to the rotating rotor blades through the use of mechanical linkages connected to a rotor disc. This introduces cyclic control where tilting the swashplate in a particular direction changes the pitch of each blade cyclically during its rotation, tilting the rotor disc and moving the helicopter forward, backward, or sideways. Additionally, raising or lowering the swashplate uniformly changes the pitch of all blades simultaneously, increasing or decreasing the overall lift. While this works very well for large helicopters, it becomes a much more challenging approach for smaller remote-controlled vehicles. The precise manufacturing required for a swashplate at that scale is much more challenging. For this project, a different approach was taken.

# Solution

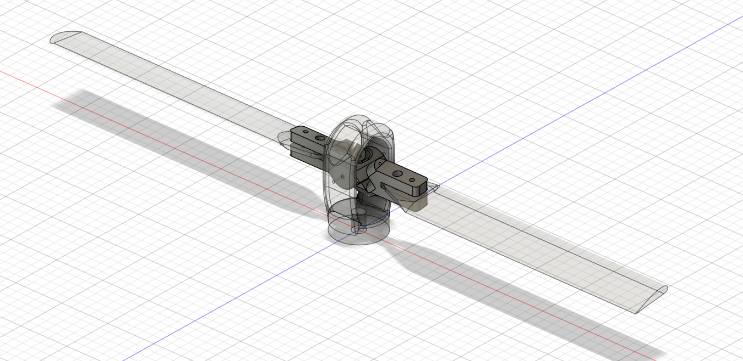


Figure 1.

Instead of using a swashplate, the duty cycle sent into the motor was modulated within a single motor rotation. Under this control scheme, the duty cycle used to drive the motor is modulated in a way similar to sine wave modulation, where the amplitude and phase of the “wave” corresponds to the magnitude and direction of pitch of the assembly. By using a simple head assembly with skewed rotor hinges, as seen in Figure 1, a lead lag relationship is seen from the individual blades. This lead lag relationship is categorized by a change in the pitch angle as the speed of the motor changes. Due to one hinge having a positive lag-pitch coefficient on one blade and a negative lag-pitch coefficient on another, lift is only generated on a single half rotation of the motor inducing pitch. This, coupled with the aforementioned duty cycle modulation, creates full cyclic control.

# Technical Approach

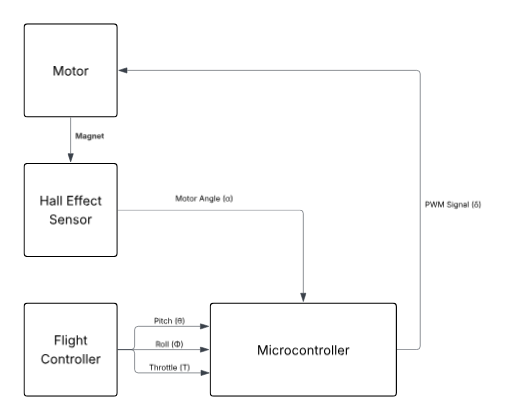


Figure 2.

The main backbone of this system is the microcontroller which is connected to the rest of the components in the system as seen in Figure 2. For this project the concept was to use an Arduino nano esp32 in order to run this system, mostly based on the size profile of the board. However this was changed to an Arduino Mega due to the requirements of at least three GPIO pins capable of hardware interrupts. These interrupts were crucial in capturing the attitude from the flight controller in a way that does not block the main program generating PWM signals.

Timing is a crucial consideration involved with this system. The PWM signal (δ) is generated based upon pitch (), roll (), motor angle (), and throttle (T) in the following equation:

(1)

Throttle, pitch, and roll are PWM signals generated from the flight controller which receives these values from a radio used by a pilot. The duty cycle must be written to the motor several times within a single rotation. The motor used, a Flite Test Radial 2218 1180kV Brushless Motor, was run at roughly 20% of maximum power using a 4-cell lithium polymer battery. If running in this configuration, a single rotation occurs once every 17 milliseconds. This is achievable with both the microcontroller clock speed and the fidelity of the AS5600 hall effect sensor. The standard pulse width for a motor uses a range between 1 and 2 milliseconds, corresponding to 0-100% motor power. This does allow for a change in duty cycle during a single motor rotation. However, this update rate is low meaning the control is not smooth and only partial control is possible. In this project, this is what was achieved.

# RESULTS

As mentioned before, cyclic control was achieved, however this only worked for very low throttle values. This is partially due to the material construction of the test stand used, as most of the parts were made of a mix of 3D printed materials and the base was made of wood. There was no damping introduced into the test stand and this resulted in critical failures at higher throttle inputs. However, a flight controller was also successfully introduced into the system in order to allow for direct pilot input to directly control pitch.

# Future Works

The largest issue in the current test stand is the materials used to construct them. Both PETG and PLA were used in the construction of the head assembly and housing for the hall effect sensor. The constant variations of the motor speed induce vibrations in the test stand which is what causes the aforementioned critical failures. Therefore, a change in the materials being used would allow the system to achieve higher thrust. Additional research into further miniaturization is also of interest, as construction of aerial vehicles at the nano scale has become prevalent in recent years.

##### References

1. J. Paulos and M Yim, “Cyclic Blade Pitch Control Without a Swashplate for Small Helicopters,” Journal of Guidance, Control and Dynamics, Jun. 2018, doi: 10.2514/1.G002683
2. K. Raymer, “An Auxiliary Actuator for Torque Modulation of a Lead-Lag Pitch Coupled Hinge Rotor

##### Appendix

/\*\*\*

\* Rotor control logic.

\* @author Michael Raymer, Wright State University

\* @author Kyle Raymer, Yellow Springs High School

\* @author Mason McDaniel, Wright State University

\* @version 1.2

\* @since 4/25/2025

\* @see - too many to mention. Almost all of this code is based on

\* Arduino library documention, examples and other sources.

\*\*/

#include <Arduino.h>

#include <Wire.h> // I2C bus

#include <AS5600.h> // Magnetic position sensor

// LOCAL PIN DEFINITIONS

#define MAIN\_PWM\_PIN 9 // ESC for main motor

#define THROTTLE\_IN\_PIN 18 // Read throttle from FC

#define PITCH\_IN\_PIN 2 // Read pitch from FC

#define ROLL\_IN\_PIN 3 // Read roll from FC

#define PWM\_ZERO 64 // Minimum power during calibration: 0 to 255 for analog\_write

#define PWM\_MIN 128 // Minimum power during flight: 0 to 255 for analog\_write

#define PWM\_MAX 252 // Maximum power during flight: 0 to 255 for analog\_write

#define MIN\_BASE\_POWER 0

#define MAX\_BASE\_POWER 100

#define PHASE\_INC 0.196349540849f

// Variables for magnet read and motor control

AMS\_5600 ams5600; // ams5600 is the magnetic angle sensor

float phase; // 0.0 to TWO\_PI

double angle; // Current rotor angle

double adjustedAngle; // Rotor angle + phase correction

byte mainPower; // Power for the main ESC: 0 to 255

float pitch;

float roll;

int Duty;

volatile int throttle;

volatile int pitchPulse;

volatile int rollPulse;

volatile unsigned long throttleRiseTime = 0;

volatile bool newThrottle = false;

volatile unsigned long pitchRiseTime = 0;

volatile bool newPitch = false;

volatile unsigned long rollRiseTime = 0;

volatile bool newRoll = false;

void setup() {

// Set up the serial output

Serial.begin(115200);

Serial.println("Initializing");

Serial.println("------------");

// Interrupts for inputs

pinMode(THROTTLE\_IN\_PIN, INPUT);

attachInterrupt(digitalPinToInterrupt(THROTTLE\_IN\_PIN), throttleISR, CHANGE);

pinMode(PITCH\_IN\_PIN, INPUT);

attachInterrupt(digitalPinToInterrupt(PITCH\_IN\_PIN), pitchISR, CHANGE);

pinMode(ROLL\_IN\_PIN, INPUT);

attachInterrupt(digitalPinToInterrupt(ROLL\_IN\_PIN), rollISR, CHANGE);

// Set up PWM pins for ESCs

pinMode(MAIN\_PWM\_PIN, OUTPUT);

// Set up I2C bus

Wire.begin(); // Initialize the I2C bus

// ESC Calibration Sequence

calibrateESC();

// Initial settings

throttle = MIN\_BASE\_POWER;

pitch = 1.0f;

phase = 2.75f; // Based on empirical testing, 3/22/24

}

void calibrateESC() {

// ESC Calibration Sequence

Serial.println("Calibrating ESC...");

// PWM\_ZERO = "no signal", 4 seconds

Serial.println("Calibrating: zero signal...");

analogWrite(MAIN\_PWM\_PIN, PWM\_ZERO);

delay(4000);

// Raise the "stick" from zero to PWM\_MAX

Serial.println("Calibrating: raising stick...");

for (byte i = PWM\_MIN; i <= PWM\_MAX; i++){

analogWrite(MAIN\_PWM\_PIN, i);

delay(10);

}

// Lower the "stick" back to min

Serial.println("Calibrating: lowering stick...");

for (byte i = PWM\_MAX; i >= PWM\_MIN; i--){

analogWrite(MAIN\_PWM\_PIN, i);

delay(10);

}

// Hold "min" for 10 seconds

Serial.println("Calibrating: minimum value...");

analogWrite(MAIN\_PWM\_PIN, PWM\_MIN);

delay(10000);

Serial.println("ESC calibration complete.\n");

}

float convertRawAngleToRadians(word newAngle){

// Raw angle is 0-4095 = 0.001533981 of a radian

float retVal = newAngle \* 0.001533981;

return retVal;

}

void throttleISR() {

if (digitalRead(THROTTLE\_IN\_PIN)) {

throttleRiseTime = micros();

} else {

unsigned long fallTime = micros();

unsigned int pulse = fallTime - throttleRiseTime;

if (pulse >= 1000 && pulse <= 2000) {

throttle = pulse;

newThrottle = true;

}

}

}

void pitchISR() {

if (digitalRead(PITCH\_IN\_PIN)) {

pitchRiseTime = micros();

} else {

unsigned long fallTime = micros();

unsigned int pulse = fallTime - pitchRiseTime;

if (pulse >= 1000 && pulse <= 2000) {

pitchPulse = pulse;

newPitch = true;

}

}

}

void rollISR() {

if (digitalRead(ROLL\_IN\_PIN)) {

rollRiseTime = micros();

} else {

unsigned long fallTime = micros();

unsigned int pulse = fallTime - rollRiseTime;

if (pulse >= 1000 && pulse <= 2000) {

rollPulse = pulse;

newRoll = true;

}

}

}

void loop() {

noInterrupts();

long localThrottle = throttle;

newThrottle = false;

int localPitch = pitchPulse;

newPitch = false;

int localRoll = rollPulse;

newRoll = false;

interrupts();

pitch = constrain((localPitch - 1000.0) / 1000.0, 0.0, 1.0);

roll = constrain((localRoll - 1000.0) / 1000.0, -1.0, 1.0);

long newThrottle = map(localThrottle, 1000, 2000, 0, 100);

Serial.print("Throttle: ");

Serial.print(newThrottle);

Serial.print(" | Pitch: ");

Serial.print(pitch);

Serial.print(" | Roll: ");

Serial.println(roll);

angle = convertRawAngleToRadians(ams5600.getRawAngle());

adjustedAngle = angle + phase;

if (adjustedAngle > TWO\_PI){

adjustedAngle -= TWO\_PI;

}

// Control law for main motor

mainPower = newThrottle + (pitch \* sin(adjustedAngle) \* newThrottle);

Duty = map(mainPower, 0, 100, PWM\_MIN, PWM\_MAX);

// Send PWM signal to ESC

analogWrite(MAIN\_PWM\_PIN, Duty);

}