THRUST VECTOR CONTROL

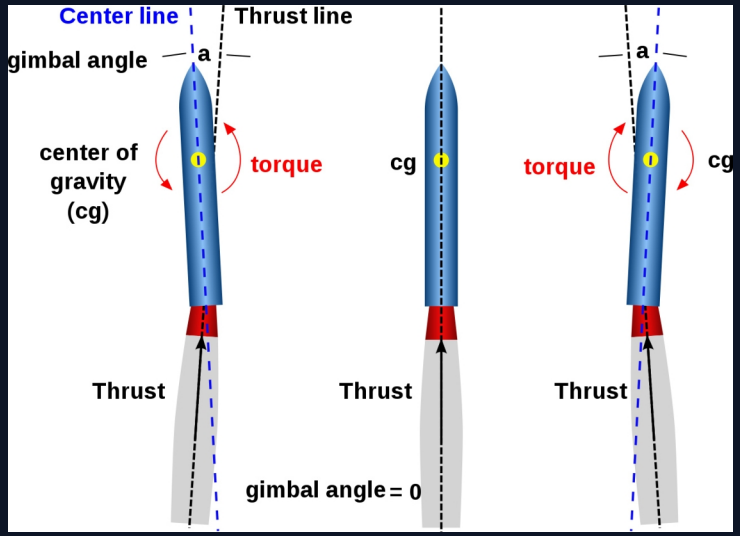
Σύστημα Σταθεροποίησης Μοντέλου Πυραύλου

Σιγάλας Μάξιμος
Τσακός Μάρκος
Πρέκας Κωνσταντίνος

01 · ΤΙ ΕΙΝΑΙ ΤΟ THRUST VECTOR CONTROL;

Ορισμός

Το TVC (Thrust Vector Control) είναι η τεχνολογία κατεύθυνσης πυραύλων που ελέγχει τη γωνία έξοδου της ώθησης. Αντί για αεροδυναμικές επιφάνειες (πτερύγια), ο κινητήρας κλίνει φυσικά — δημιουργώντας ροπή που διορθώνει την τροχιά.



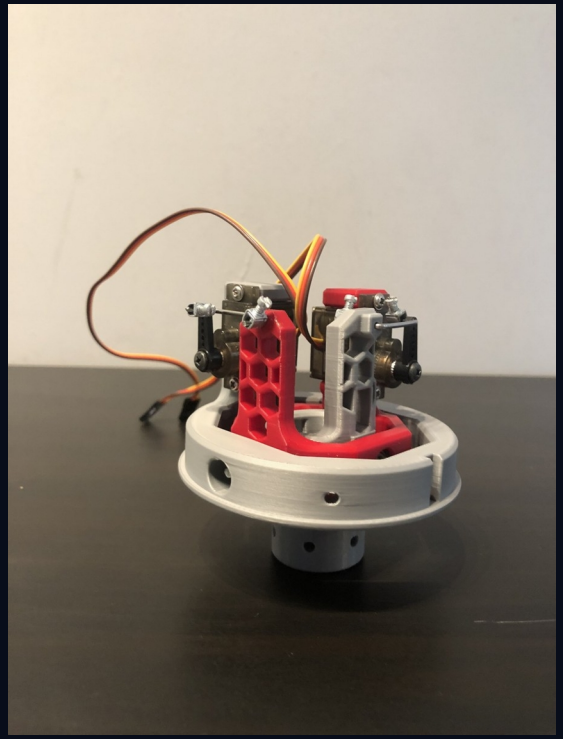
Κύριο Πλεονέκτημα

Ελέγχει την κλήση κατά τη φάση ώθησης ανεξάρτητα από την αεροδυναμική χωρίς την χρήση πτερυγίων

SpaceX, NASA, ESA

Falcon 9, Saturn V, Ariane 5 χρησιμοποιούν TVC ως κύριο σύστημα ελέγχου

02 · ΜΗΧΑΝΙΣΜΟΣ GIMBAL



2× MG90S Micro Servo

S1 (κόκκινο) → Pitch axis
S2 (γκρι) → Yaw axis
Επειδή εχουν διαφορετική γεωμετρία απαιτούνται διαφορετικά steps/° ανά άξονα

3D Εκτυπωμένο Πλαίσιο

Εκτυπώθεικε με πλαστικό PLA (πολυγαλακτικό οξύ) και χρησιμοποιήθικε Gyroid infill για επιπλεον δυναμη στις δονήσης

Εύρος Κίνησης

±15° Pitch (μπρος-πίσω)
±15° Yaw (δεξιά-αριστερά)

03 · ΗΛΕΚΤΡΟΝΙΚΑ & ΥΛΙΚΟ — Τα Εξαρτήματα

Electronics Bay (ενεργό)

Πίσω όψη — χειροποίητη καλωδίωση

Χαρτογράφηση Pins:

`MPU6050 SDA→A4 SCL→A5 | Servo: S1→D9 S2→D10 | LiPo: VIN(7.4V) | 3.3V`

Arduino Nano

Επεξεργαστής: ATmega328P @ 16 MHz
Μνήμη: 32KB Flash · 2KB SRAM
Επικοινωνία: I2C (A4/A5) + PWM
Loop: 100 Hz (κάθε 10ms)

MPU-6050 IMU

3-axis Γυροσκόπιο + 3-axis Επιταχυνσιόμετρο
Ευαισθησία: ±500°/s
Χρησημοποιήτε για ανίχνευση γωνίας pitch & roll
Τροφοδοσία: 3.3V (μέσω Arduino)

2S LiPo 7.4V

Τροφοδοτεί: Arduino (VIN) και σερβοκινητήρες
Σύνδεση: XT30 connector

2× MG90S Servo

Τροφοδοτούνται με 6V
S1 → D9 (Pitch)
S2 → D10 (Yaw)

04 · ΑΛΓΟΡΙΘΜΟΣ: Συμπληρωματικό Φίλτρο (Complementary Filter)

Το Πρόβλημα — Γιατί Χρειαζόμαστε Φίλτρο;

Γυροσκόπιο: Πολύ γρήγορο, μετρά γωνιακή ταχύτητα (°/s) ακριβώς. ΑΛΛΑ:
→ Drift: Μικρά σφάλματα συσσωρεύονται με τον χρόνο → η γωνία «περπατά» αργά

Επιταχυνσιόμετρο: Δίνει απόλυτη γωνία από τη βαρύτητα. ΑΛΛΑ:
→ Noise: Τεράστιος θόρυβος κατά τις δονήσεις του κινητήρα

Η Λύση — Συνδυασμός και των Δύο

Το Συμπληρωματικό Φίλτρο συνδυάζει:
- 98% γυροσκόπιο → για γρήγορες κινήσεις (CF = 0.98)
- 2% επιταχυνσιόμετρο → για μακροπρόθεσμη διόρθωση

Αποτέλεσμα: Γρήγορη απόκριση χωρίς drift!

Ο Μαθηματικός Τύπος:

$$angle = CF \times (angle + \omega \times dt) + (1 - CF) \times \theta_{accel}$$

```
angle     → η τρέχουσα εκτίμηση γωνίας (output)
ω×dt      → ολοκλήρωση γυροσκοπίου (gyro_rate × χρόνος)
θ_accel   → γωνία από επιταχυνσιόμετρο
CF=0.98   → συντελεστής εμπιστοσύνης γυροσκοπίου
```

`complementary_filter.ino — κάθε 10ms`

```
// Χρόνος από το τελευταίο loop
float dt = (now - lastTime) / 1000.0;
lastTime = now;

// Ανάγνωση IMU
mpu.update();
float ax = mpu.getAngleX(); // accel
float ay = mpu.getAngleY();
float gx = mpu.getGyroX(); // °/s
float gy = mpu.getGyroY();

// Συμπληρωματικό Φίλτρο
const float CF = 0.98;
pitch = CF*(pitch + gx*dt)
      + (1-CF)*ax;
roll  = CF*(roll  + gy*dt)
      + (1-CF)*ay;
```

Ροή Σήματος

05 · ΑΛΓΟΡΙΘΜΟΣ: Ελεγκτής PID — Proportional · Integral · Derivative

Ο PID ελεγκτής υπολογίζει πόσο πρέπει να κινηθεί το gimbal ώστε ο πύραυλος να επιστρέψει στον κατακόρυφο άξονα. Παίρνει ως είσοδο το σφάλμα (απόκλιση από 0°) και δίνει έξοδο στα servo.

P — Proportional — Αναλογικό

Αντιδρά ΑΜΕΣΩΣ στο τρέχον σφάλμα.
Όσο μεγαλύτερη η κλίση, τόσο μεγαλύτερη η διόρθωση.
Κίνδυνος: Αν Kp πολύ μεγάλο → Δημιουργει ταλαντώσεις
out_P = Kp × error

```
float errP = pitch;  // απόκλιση από 0°
float outP = Kp * errP;
```

I — Integral — Ολοκλήρομα

Συσσωρεύει το σφάλμα με τον χρόνο.
Διορθώνει μόνιμες παρεκκλίσεις (steady-state error).
Κίνδυνος: Windup — το άθροισμα εκτοξεύεται
out_I = Ki × Σ(error × dt)

```
sumP += errP * dt;  // ολοκλήρωση
float outI = Ki * sumP;
```

D — Derivative — Παράγωγη

Μετρά τον ΡΥΘΜΟ αλλαγής του σφάλματος.
«Προβλέπει» την κίνηση
Κίνδυνος: Αν Kd πολύ μεγάλο → Αυξάνετε ο θόρυβος
out_D = Kd × Δerror/Δt

```
float outD = Kd*(errP - prevP)/dt;
prevP = errP;
```

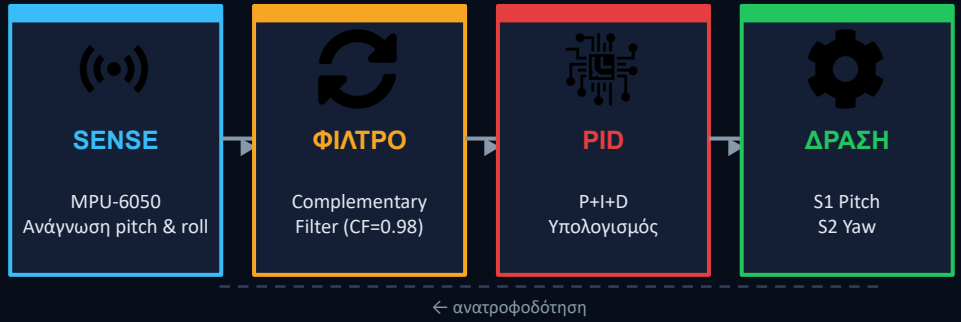
Τελική Έξοδος στα Servo:

totalOutput = Kp×error + Ki×Σ(error×dt) + Kd×(Δerror/Δt)

```
S1_pos = S1_CENTER + totalOutput_pitch × STEPS_PER_DEG_S1
S2_pos = S2_CENTER + totalOutput_roll  × STEPS_PER_DEG_S2
```

06 · ΠΩΣ ΟΛΑ ΛΕΙΤΟΥΡΓΟΥΝ ΜΑΖΙ — Ο Βρόχος Ελέγχου

Βρόχος Ελέγχου (κάθε 10ms = 100 Hz)



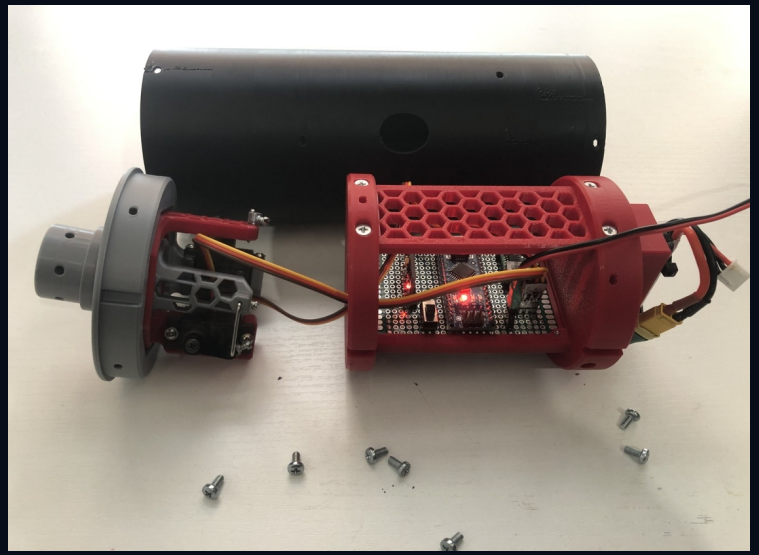
loop() — Ο Πλήρης Βρόχος @ 100 Hz

```
void loop() {
  unsigned long now = millis();
  float dt = (now-lastT)/1000.0;
  if (dt < 0.01) return; // 100Hz limit
  lastT = now;

  mpu.update();            // 1. SENSE
  pitch = CF*(pitch+gx*dt)+(1-CF)*ax;// 2.FILTER
  roll  = CF*(roll +gy*dt)+(1-CF)*ay;

  outP = Kp*pitch + Ki*sumP + Kd*dP; // 3.PID
  outR = Kp*roll  + Ki*sumR + Kd*dR;

  s1.write(S1_C + outP*STEPS_P);  // 4.ACT
  s2.write(S2_C + outR*STEPS_R);
}
```



Πλήρες σύστημα — exploded view

Αρχικοποίηση (setup())

MPU6050 I2C init · Servo attach D9/D10 · Settling loop 200 iterations

Βαθμονόμηση Servo

S1: 5.2 steps/° (pitch) · S2: 3.8 steps/° (yaw)

Παράμετροι PID

Kp=1.8 Ki=0.05 Kd=0.12