- **Electronic Speed Controllers (ESCs):**
    - Devices used to control the speed and direction of thrusters.
    - They translate PWM signals into motor speed.
    - Typical PWM range:
        - **1000 µs:** Minimum speed (reverse if supported).
        - **1500 µs:** Neutral (stationary).
        - **2000 µs:** Maximum speed (forward).
- **Thrusters:**
    - Driven by brushless motors.
    - Controlled individually or in pairs for directional control.

ESC and Thruster Integration

- **ESC Setup:**
    - ESCs require calibration to match PWM ranges. This is usually done once at startup.
    - Each ESC is assigned to a specific thruster.
- **PWM Signal Mapping:**
    - The code calculates PWM values dynamically based on input commands and control algorithms (e.g., joystick commands, and PID output).

Control Modes

1. **Individual Thruster Control:** Each thruster can be controlled independently.
2. **Directional Control:** PWM signals are combined to achieve specific movements (e.g., forward, backward, rotate).

Code :

```
Global Variables:

Define constant THRUSTER_PWM_NEUTRAL = 1500

Define constant THRUSTER_PWM_MIN = 1000

Define constant THRUSTER_PWM_MAX = 2000


Initialize PWM variables for all thrusters:

  Thruster_intLeftFrontPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intLeftBackPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intRightFrontPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intRightBackPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intUpFrontPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intUpBackPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intDownFrontPWM = THRUSTER_PWM_NEUTRAL

  Thruster_intDownBackPWM = THRUSTER_PWM_NEUTRAL
```

Function: Thruster_voidParseCommand

Purpose: Parse the incoming command string and update the PWM values for each thruster.

```
FUNCTION Thruster_voidParseCommand(Copy_strCommand)
    FOR each thruster label ('A' to 'H'):
        Extract PWM value using Thruster_intGetPWMValue(Copy_strCommand, thruster_label)
        Update the corresponding PWM variable:
            IF label = 'A': Thruster_intLeftFrontPWM = Extracted PWM value
            IF label = 'B': Thruster_intLeftBackPWM = Extracted PWM value
            IF label = 'C': Thruster_intRightFrontPWM = Extracted PWM value
            IF label = 'D': Thruster_intRightBackPWM = Extracted PWM value
            IF label = 'E': Thruster_intUpFrontPWM = Extracted PWM value
            IF label = 'F': Thruster_intUpBackPWM = Extracted PWM value
            IF label = 'G': Thruster_intDownFrontPWM = Extracted PWM value
            IF label = 'H': Thruster_intDownBackPWM = Extracted PWM value
    END FOR
END FUNCTION
```

Function: Thruster_intGetPWMValue

Purpose: Extract the PWM value for a specific thruster from the command string.

```
FUNCTION Thruster_intGetPWMValue(Copy_strCommand, Copy_charLabel)
    StartIndex = Find position of Copy_charLabel in Copy_strCommand
    EndIndex = Find next space or end of string after StartIndex
    PWMValue = Convert substring between StartIndex and EndIndex to integer
    RETURN PWMValue
END FUNCTION
```

Function: Thruster_voidApplyPWM

Purpose: Apply the calculated PWM values to the ESCs controlling the thrusters.

```
FUNCTION Thruster_voidApplyPWM()
    FOR each thruster servo:
        IF servo = Thruster_SERLeftFront:
            Send Thruster_intLeftFrontPWM to ESC
        IF servo = Thruster_SERLeftBack:
            Send Thruster_intLeftBackPWM to ESC
        IF servo = Thruster_SERRightFront:
            Send Thruster_intRightFrontPWM to ESC
        IF servo = Thruster_SERRightBack:
            Send Thruster_intRightBackPWM to ESC
        IF servo = Thruster_SERUpFront:
            Send Thruster_intUpFrontPWM to ESC
        IF servo = Thruster_SERUpBack:
            Send Thruster_intUpBackPWM to ESC
        IF servo = Thruster_SERDownFront:
            Send Thruster_intDownFrontPWM to ESC
        IF servo = Thruster_SERDownBack:
            Send Thruster_intDownBackPWM to ESC
    END FOR
END FUNCTION
```

Joystick Input:

- Directly read the X and Y analog values from the joystick (joystickX on A0, joystickY on A1).

Mapping:

- Map the raw joystick values (0–1023) to servo angles (0–180).

Servo Movement:

- Write the mapped angles to the corresponding servo motors.

Reset:

- If the button is pressed, reset both servos to their default angles (90°).

```
FUNCTION setup()
    Attach servoX to pin 6
    Attach servoY to pin 7
    Set buttonPin as INPUT with pull-up
    Set initial servo positions to 90° (default)
    Start Serial communication
    Print initialization message
END FUNCTION

FUNCTION loop()
    Read analog joystick values:
        joystickValX = analogRead(joystickX)
        joystickValY = analogRead(joystickY)

    Map joystick values (0–1023) to servo angles (0–180):
        angleX = map(joystickValX, 0, 1023, minAngle, maxAngle)
        angleY = map(joystickValY, 0, 1023, minAngle, maxAngle)

    Write angles to servos:
        servoX.write(angleX)
        servoY.write(angleY)

    Print angles to Serial Monitor
    IF reset button is pressed:
        Call resetPosition()

    Delay 100ms
END FUNCTION
```