PseudoCode

```
// Include necessary libraries
INCLUDE "SSD1306"
INCLUDE "DS18B20"
INCLUDE "RotaryEncoder"
INCLUDE "config"
INCLUDE "pid"

// Initialize display, temperature sensor, and encoder
INITIALIZE display AS SSD1306
INITIALIZE sensor AS DS18B20 USING DS18B20_PIN
INITIALIZE encoder AS RotaryEncoder USING Encoder_CLK, Encoder_DT, Encoder_SW

// Define temperature settings
SET minTemp TO 10
SET maxTemp TO 25
SET currentTemp TO 0 // Current process value
SET desiredTemp TO 18 // Initial desired temperature
SET position TO 18 // Initial position for desired temperature
SET lastPosition TO encoder.getPosition()
SET last_CLK TO 0
SET fn TO (fc / fs) // Calculate frequency ratio
SET alpha TO 0.0
SET error_filtered_previous TO 0
SET last_integral TO 0
SET time_stamp_previous TO 0

FUNCTION setup()
        INITIALIZE encoder
        INITIALIZE display

        IF NOT sensor.begin() THEN
        PRINT "Could not find DS18B20 sensor."
        WHILE TRUE // Halt execution
        END WHILE
        END IF

        SET pinMode for pumpA, pumpB, pumpPWM, and peltierControl TO OUTPUT

        display.clearDisplay()
        display.drawTable() // Draw permanent screen components
        display.display()

        SET time_stamp_previous TO current time
        SET alpha TO calculateAlphaEMAFilter(fn) // Calculate alpha for EMA filter
```

```
END FUNCTION

FUNCTION loop()
	WHILE true
	// Update rotary encoder state
	encoder.update()
	SET position += encoder.getPosition()

	// Wrap around if exceeded bounds
	IF position > maxTemp THEN
	SET position TO minTemp
	ELSE IF position < minTemp THEN
	SET position TO maxTemp
	END IF

	PRINT "Encoder position: " + position

	IF encoder.checkButtonPressDuration() > 2000 THEN
	PRINT "Finalise key pressed!"
	SET desiredTemp TO position // Update desired temperature based on encoder
position
	END IF

	CALL pidFunction() // Execute PID control function
	DELAY(100) // Delay for stability

	// Read temperature from sensor
	SET tempReading TO sensor.getTemperatureC()
	SET currentTemp TO tempReading

	IF currentTemp != -127.0 THEN
	PRINT "Temperature: " + currentTemp + " °C"
	display.drawText(currentTemp, 20, 28) // Display current temperature
	ELSE
	PRINT "Error reading temperature"
	END IF

	display.drawText(desiredTemp, 80, 28) // Display desired temperature
	display.display()
	DELAY(100) // Delay before next loop iteration
	END WHILE
END FUNCTION

FUNCTION pidFunction()
	SET error TO desiredTemp - currentTemp // Calculate error
	SET current_time TO current time
	SET deltaTime TO (current_time - time_stamp_previous) / 1000.0 // Convert to
seconds
```

```
        // PID calculations
        SET integral TO last_integral + (error * deltaTime)
        SET derivative TO (error - error_filtered_previous) / deltaTime
        SET output TO (Kp * error) + (Ki * integral) + (Kd * derivative) // PID formula

        // Update control values for pump and Peltier
        SET pumpControlValue TO constrain(output, 0, 255) // PWM limits for pump
        SET peltierControlValue TO constrain(output, 0, 255) // PWM limits for Peltier

        PRINT "Pump Control Value: " + pumpControlValue
        PRINT "Peltier Control Value: " + peltierControlValue

        digitalWrite(pumpA, HIGH) // Activate pumpA
        digitalWrite(pumpB, LOW) // Deactivate pumpB
        analogWrite(pumpPWM, pumpControlValue) // PWM control for pump
        analogWrite(peltierControl, peltierControlValue) // Control Peltier

        // Update variables for next loop
        SET error_filtered_previous TO error
        SET last_integral TO integral
        SET time_stamp_previous TO current_time // Update timestamp
END FUNCTION

FUNCTION calculateAlphaEMAFilter(frequency)
        // Example EMA filter calculation logic (update as necessary)
        SET alpha TO 2.0 / (frequency + 1) // Adjust formula as needed
        RETURN alpha
END FUNCTION
```