PID Control with FreeRTOS

03/11/2024

Today's Topic:

PID Control With FreeRTOS

AGENDA

1. Welcome and introductions
2. Agenda
3. What is feedback and control
4. Drill down explanation in context of reference setup
5. Introduce reference hardware, software and/or tools
6. Live demonstration
7. Closing thoughts and considerations
8. Looking forward
9. Q&A
10. Outro

Today's Topic: Control with FreeRTOS—**Introduction**

What to expect from today's session

- ➔ An introduction to control theory and understanding of feedback and control.
- ➔ A brief introduction to the core concepts of feedback and how to use them for control
- ➔ Using FreeRTOS to combine all these so that we can focus on our application rather than tooling.

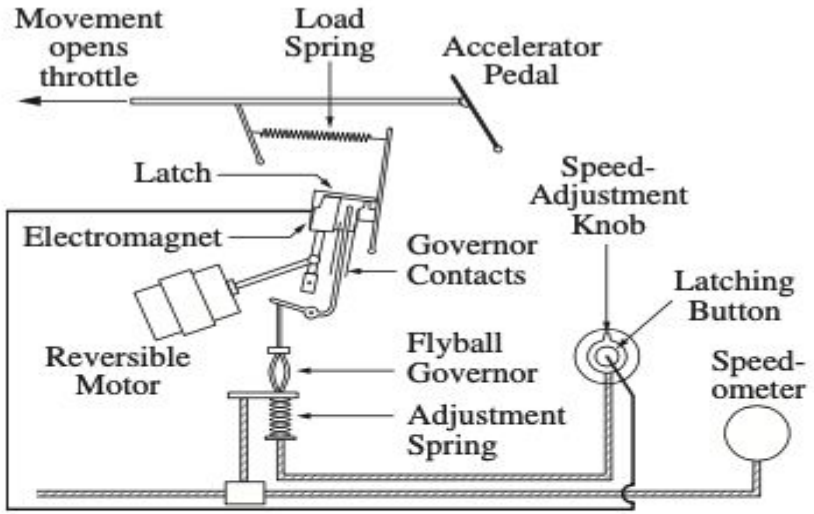
Today's Topic: Control with FreeRTOS/

Why do we need control

Today's Topic: Control with FreeRTOS/

Why do we need control

(a) Honeywell thermostat, 1953



(b) Chrysler cruise control, 1958

Figure 1.4: Early control devices. (a) Honeywell T87 thermostat originally introduced in 1953. The thermostat controls whether a heater is turned on by comparing the current temperature in a room to a desired value that is set using a dial. (b) Chrysler cruise control system introduced in the 1958 Chrysler Imperial [Row58]. A centrifugal governor is used to detect the speed of the vehicle and actuate the throttle. The reference speed is specified through an adjustment spring. (Left figure courtesy of Honeywell International, Inc.)

Today's Topic: Control with FreeRTOS

Why do we need control

Control is essential for ensuring stability, precision, and responsiveness in dynamic systems, enabling them to adapt in real-time to changing conditions and maintain optimal performance.

The need for control:

- allows precise management over dynamic behaviors,
- ensuring that systems operate within desired parameters despite external disturbances or changing demands.
- automated processes can respond effectively to real-time feedback, correcting deviations and maintaining stability.

Today's Topic: Control with FreeRTOS

What is control

Control Involves determining how much and how quickly to apply correction (aka feedback) to achieve an optimum set point.

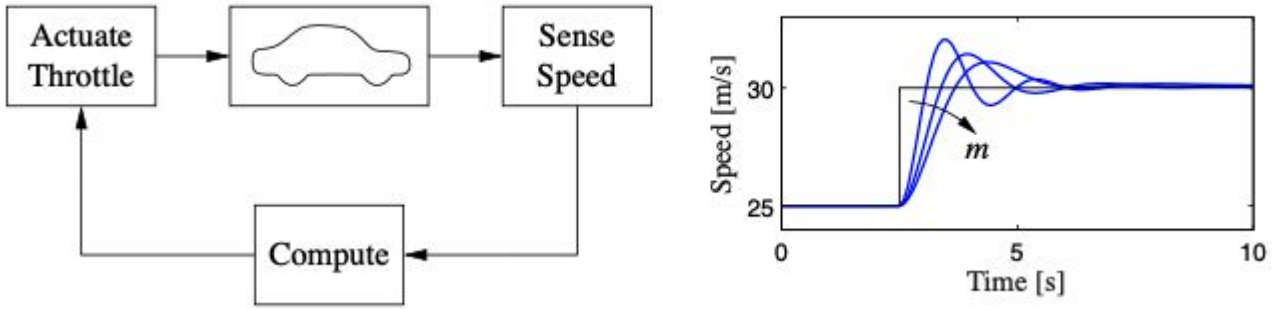
Control:

A modern controller senses the operation of a system, compares it against the desired behavior, computes corrective actions based on a model of the system's response to external inputs and actuates the system to effect the desired change. This basic feedback loop of sensing, computation and actuation is the central concept in control.

Much of the early development of control was driven by the generation and distribution of electrical power. Control is mission critical for power systems, and there are many control loops in individual power stations.

Today's Topic: Control with FreeRTOS

The principle of feedback



The principle of feedback is simple: Base correcting actions on the difference between desired and actual performance.

The use of feedback has often resulted in vast improvements in system capability. These improvements have sometimes been revolutionary.

Today's Topic: Control with FreeRTOS/

Forms of control — On/Off

The idea of feedback to make corrective actions based on the difference between the desired and the actual values of a quantity can be implemented in many different ways

Forms Control:

- On- Off Control - SImple on off, if it's dark put on the light, if its light put off the lights.
- This is fine for your house and thermostat since it should float around a desired set point
- However these leads to oscillation, and for industrial and other process the results would be the results would be less than desirable, you want cruise control to go at 80 not hover 20% above and below 80

Today's Topic: Control with FreeRTOS/

On/Off Control

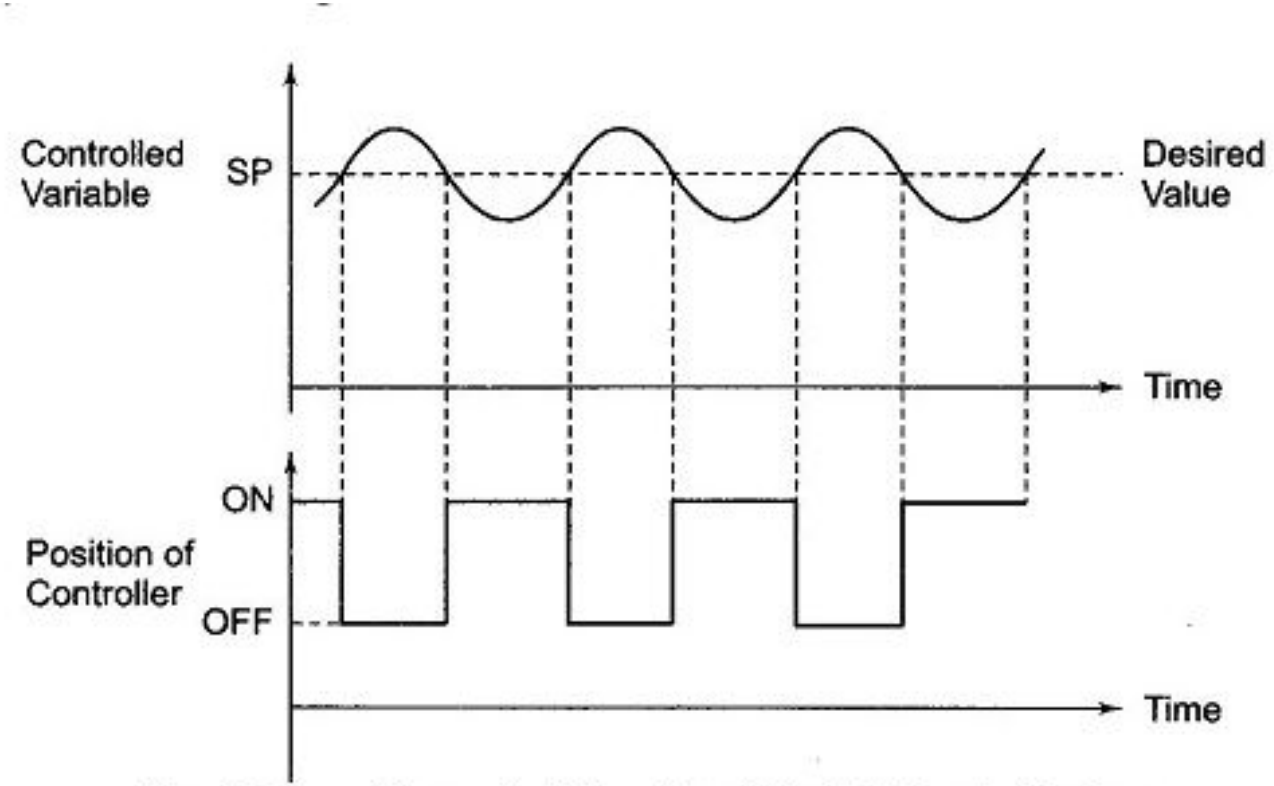


Fig. 21.3 Characteristics of an ON–OFF Control Action

Today's Topic: Control with FreeRTOS/

Forms of control—Proportional control

Here we apply correction (feedback) that us proportional to the error signal.

The error signal here is the difference between a desired setpoint and the measured value.

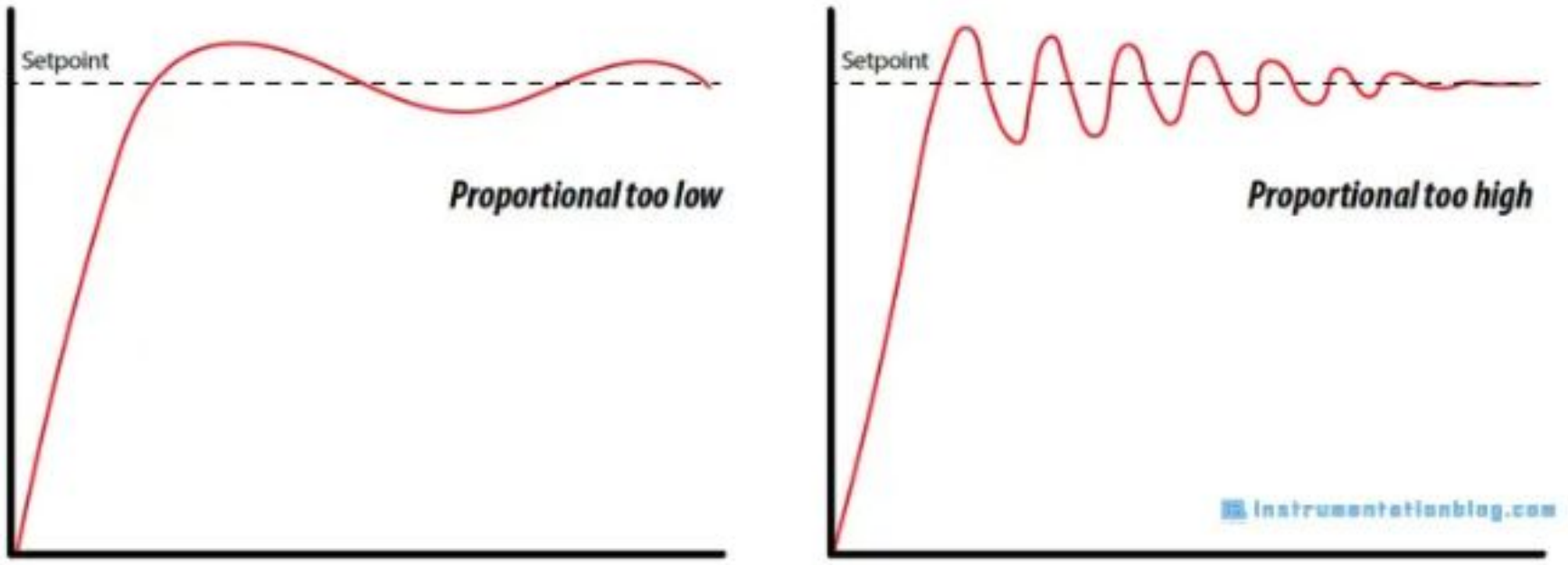
Proportional Control:

- It responds immediately with an instant reaction to errors, as the controller adjusts output in direct proportion to the size of the error.
- We can tune this system but that can cause instability if the errors are too high.
- Beware of the steady state error, a small enough error may remain to keep the controller active

Good for were a fast and straightforward feedback is enough such as basic motor control where staying close enough to a value is file, such as a RPMs

Today's Topic: Control with FreeRTOS/

Proportional control



Proportional control and what happens when the error is too low or too high

Today's Topic: Control with FreeRTOS/

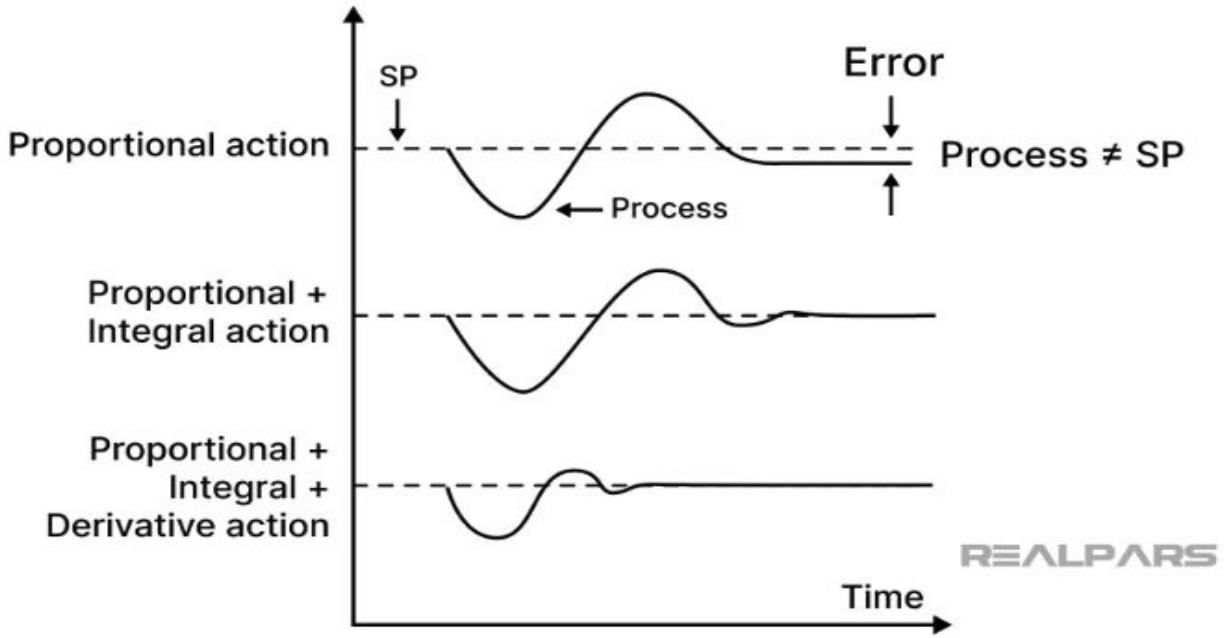
PID Control

What happens when we need even smoother signal. Say we have a motor that to run at an extremely accurate RPM? Where even the state state error in proportional control would be catastrophic.

- By accumulating the errors over time we can apply a corrective action that grows until the error is Zero. However this makes the system slower and introduces overshoots and oscillations. An accumulation of error is called an integral.
- If we observe the current rate of change to the process variable and try to balance that based on the rate we are observing, we can prevent overshoot or oscillations by damping them. A derivative is a rate of change in a quality, here the error.
- Both of this are based on a proportion, which is the amount of the current error and hence we have **Proportional-Integral-Derivative** control.

Today's Topic: Control with FreeRTOS/

PID Control



Today’s Topic: Control with FreeRTOS/

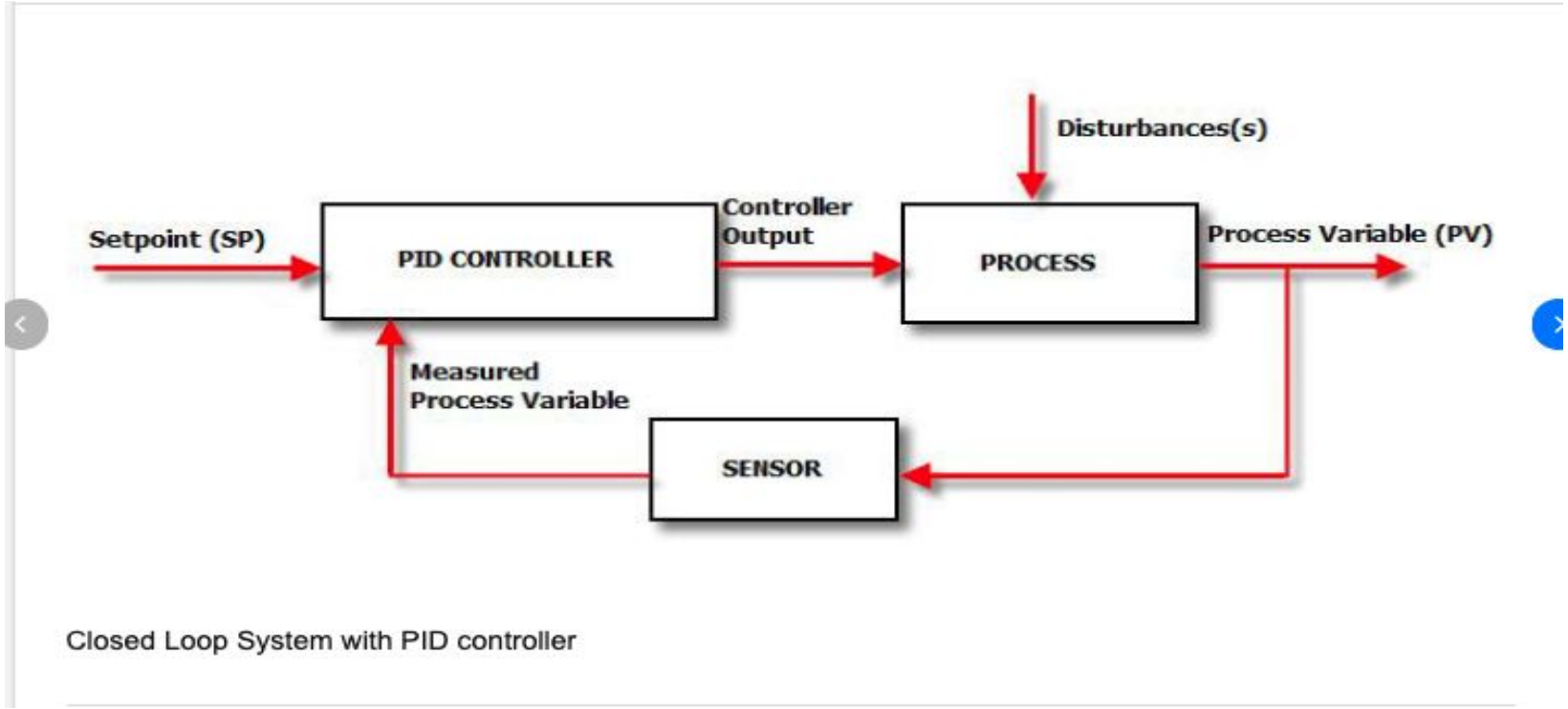
Application Example: *PWM fuel pump controller*

The controller adjusts the fuel pump's speed based on real-time feedback from a pressure sensor to maintain optimal fuel pressure under varying engine demands. By employing pulse-width modulation (PWM), the system precisely regulates pump power, reducing energy consumption and extending pump life.

Lets us explore how to implement closed-loop control, where sensor feedback continuously informs PWM adjustments, providing stable, efficient fuel delivery.

Today's Topic: Control with FreeRTOS/

PID Control Process



Closed Loop System with PID controller

Today's Topic: Control with FreeRTOS/

Why FreeRTOS

FreeRTOS is an excellent OS for control applications, let's look at why.

1. Real-Time Performance: Deterministic scheduling for timely responses.
2. Lightweight: Efficient use of limited resources.
3. Task Management: Supports multiple tasks with prioritization.
4. Inter-task Communication: Queues and semaphores for data exchange.
5. Portability: Works across various hardware platforms.
6. Configurable: Customizable for specific application needs.
7. Low Latency: Minimal context switching delays.
8. Safety and Reliability: Designed for safety-critical applications.

Today's Topic: Control with FreeRTOS/

A FreeRTOS Task for PID Control

By using a number of freeRTOS tasks, we can implement PID control very effectively.

Lets focus on the task doing the heavy lifting and understand that.

- Periodically our task will read the pressure sensor to determine if the target pressure is met
- Within this task we will use some variables to set the pumps pwm signal
- When the task reads te sensor (process variable) it uses it to determine the error
- The error is the difference between the set point and the current reading. A positive arrow indicates pressure is too low and a negative indicates pressure is too high.
- Use the error to calculate the PID output, the out put of this gives us the desired adjustment of the pwm duty.

ET VOILA! PID

Today's Topic: Control with FreeRTOS

Tools/hardware

IDE: VS Code, Combined with the cortex debug extension

Evaluation Kit: STM32F407G

OS: FreeRTOS

CODE:

https://github.com/DevHeadsCommunity/FreeRTOS_samples/pull/1

Today's Topic: Control with FreeRTOS/

Steps to implement PID in FreeRTOS

1. Set Up the GPIO Pin: Enable GPIO clock, configure pin for PWM output, set speed, and select alternate function.

2. Initialize Timer for PWM: Enable timer clock, configure for PWM mode, set ARR for frequency, and initialize duty cycle in CCR.

3. Define PID Controller Structure: Create a structure for PID gains, setpoint, integral, and previous error.

4. Implement PID Calculation Function: Develop a function to compute PID output based on measured value and calculate error components.

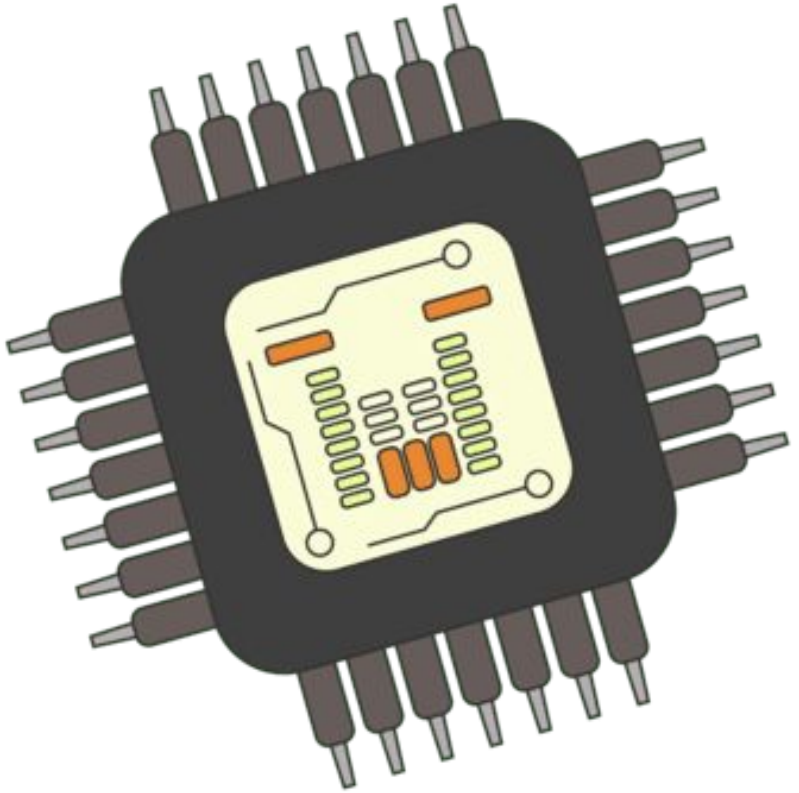
5. Control Loop Function: Create a function to read sensor value, call PID calculation, and update PWM duty cycle.

6. Create FreeRTOS Task: Implement a FreeRTOS task to read pressure, execute control loop, and include a delay.

7. Task Creation: Use `xTaskCreate` to start the FreeRTOS task with appropriate priority and stack size.

Today's Topic: DSP with FreeRTOS

Coding Example



THANKS FOR BEING A DEVHEAD!

- https://discord.gg/DevHeads