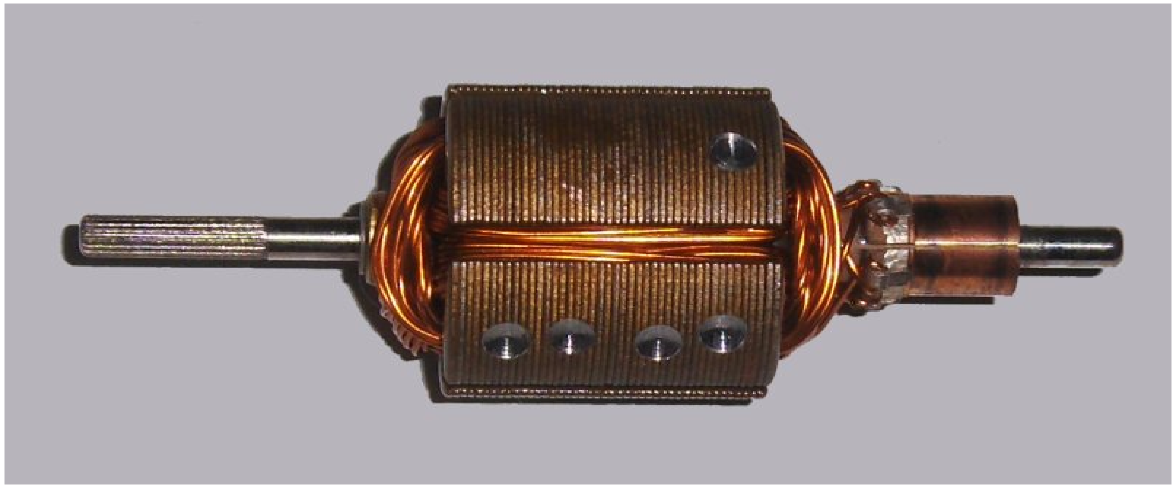
Rapid PID Controller Prototyping for Brushed DC Motors



Alex Cortner

Problem:

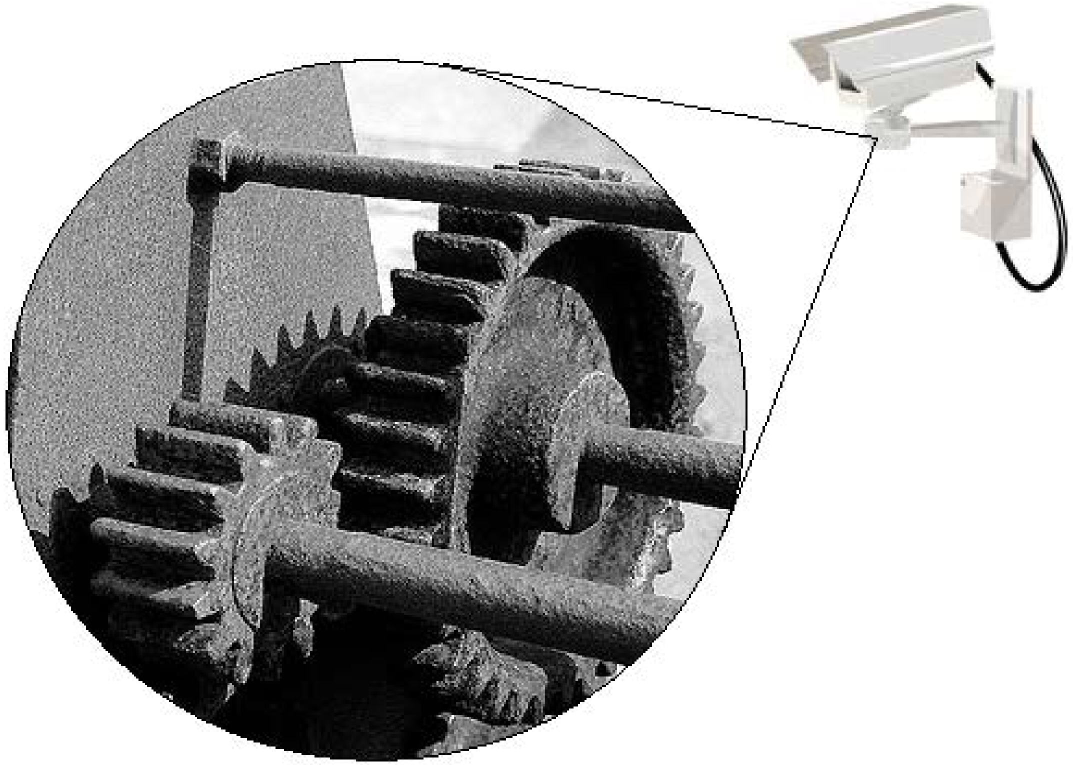
Aging **analog** control systems for brushed DC motors

Example:

3-axis gimbal cameras

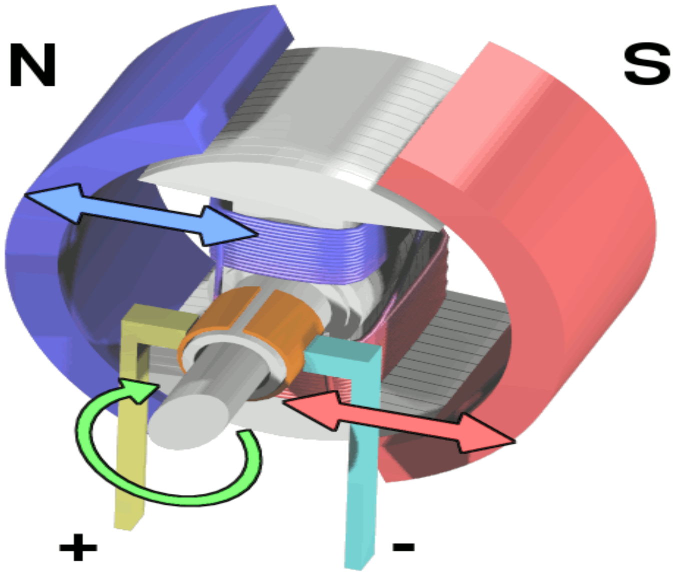
(3 analog

motor controllers)

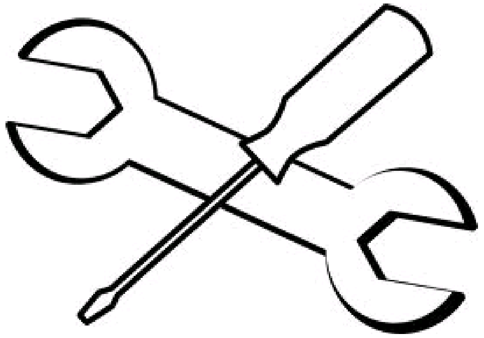


Brush-type DC Motor Overview

- *Stator: stationary part*
- *Armature: a coil with rotation-inducing currents*
- Armature mounted on the rotor – it <u>rotates</u>!
- Split-ring contacts supply current – alternates the DC
- Stator: permanent magnet or static electromagnet



Brush-type DC Motor Characteristics



- Often high maintenance
- Feedback electronics obsolete, expensive to repair
- Controls often fairly customized
- Better to replace controls than to repair

Replacement of Feedback Control System

- Prefer cheap and generic -- *Idea: PID controller!*
- (Proportional, Integral, Derivative)
- PID is the most common feedback controller

$$\mathrm{MV(t)} = P_{\mathrm{out}} + I_{\mathrm{out}} + D_{\mathrm{out}}$$

- MV = the “manipulated variable” being controlled
- P, I, & D: calculated terms that modify the error signal
- Adjustment of these terms is called “<u>tuning</u>”

PID Controller Implementation

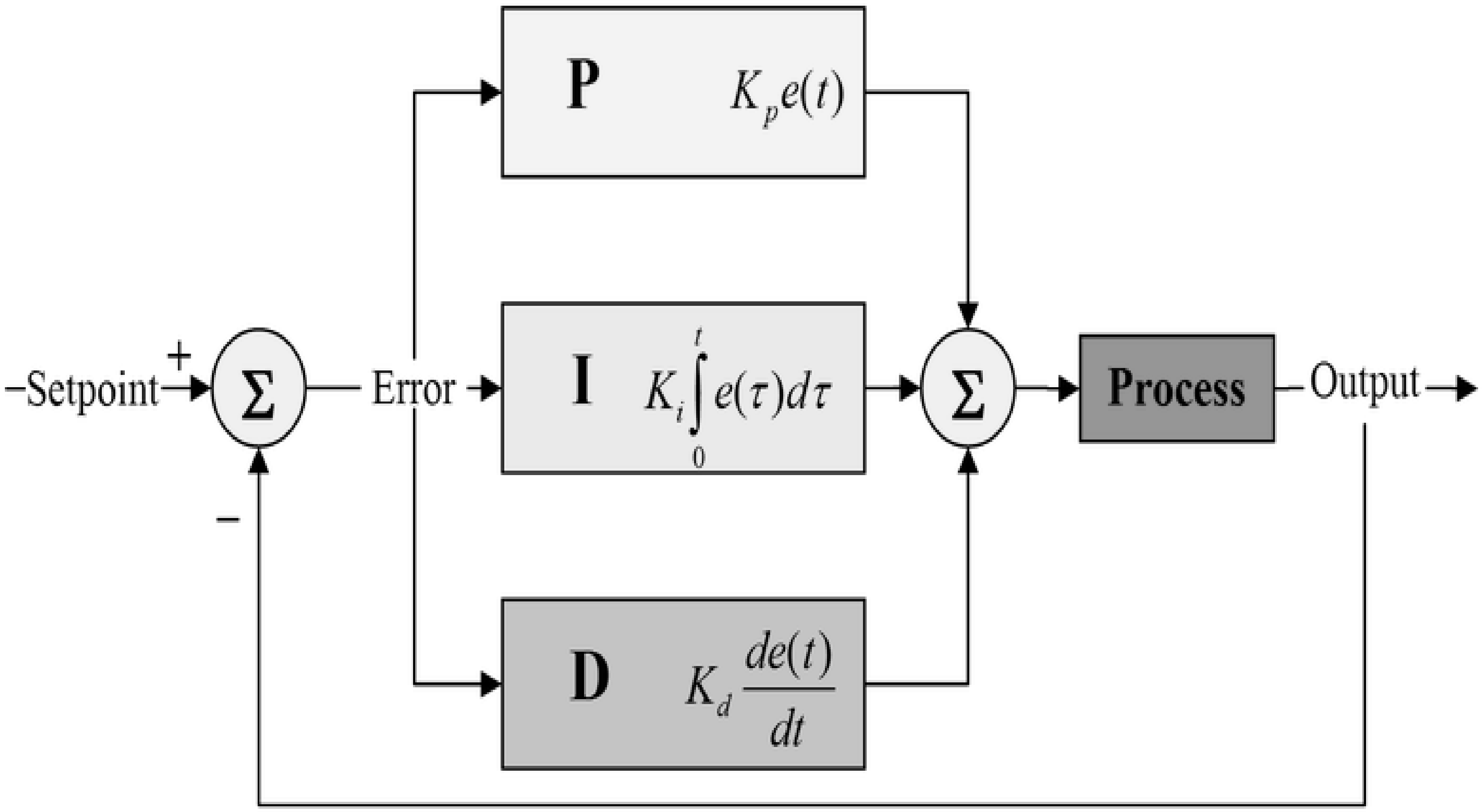
- Can be analog, but digital is better (maybe cheaper)

Why?

- Digital = programmable, so fast / easy prototyping
- Offers specificity (customizability)
- Enhanced performance
- Easily adaptable to existing equipment

Analog DIGITAL

Background info: The PID Controller



The “P”

P (*proportional*)

$$P_{\text{out}} = K_p e(t)$$

- Based on present error
- (Digital: present input sample only)
- Term that contributes the most to the output
- Too much causes overshoot, instabilities, oscillations

The “I”

I (*integral*)

$$I_{\text{out}} = K_i \int_0^t e(\tau)\, d\tau$$

- Based on accumulation of <u>past</u> errors
- (Digital: <u>present</u> & <u>last</u> input samples)
- The “**I**ntrospective” term!
- Too much causes overshoot, instabilities, oscillations

The "D"

D (***derivative***)

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

- Prediction of future error based on current rate of change
- (Digital: uses the last 10 input samples)
- Can dampen overcompensation (stabilizes)
- Helps reduce overshoot arising from P & I terms

How to implement PID controller digitally?

Digital Implementation with FPGA

FPGA offers:

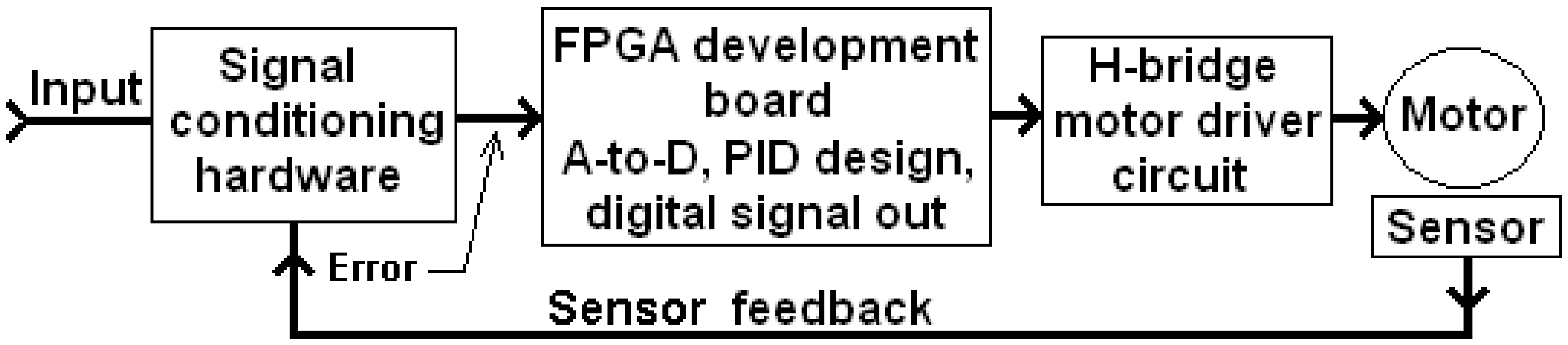
- Off-the-shelf design platform
- Customized through programming
- Can use PID designs already written; just "**tune**" to fit
- PID controller for each motor implemented in a 'soft core' on the FPGA
- kHz controllers are easy for a MHz FPGA – can implement several on one

3-Axis Camera Gimbal Example

Implementation of the 3-axis gimbal controller:

- Xilinx Spartan 3E FPGA development board
- (Free) Xilinx PicoBlaze 8-bit u-controller/soft-core (one for each motor)
- Use Xilinx KCPSM assembler to auto-generate VHDL based on assembly language PID design

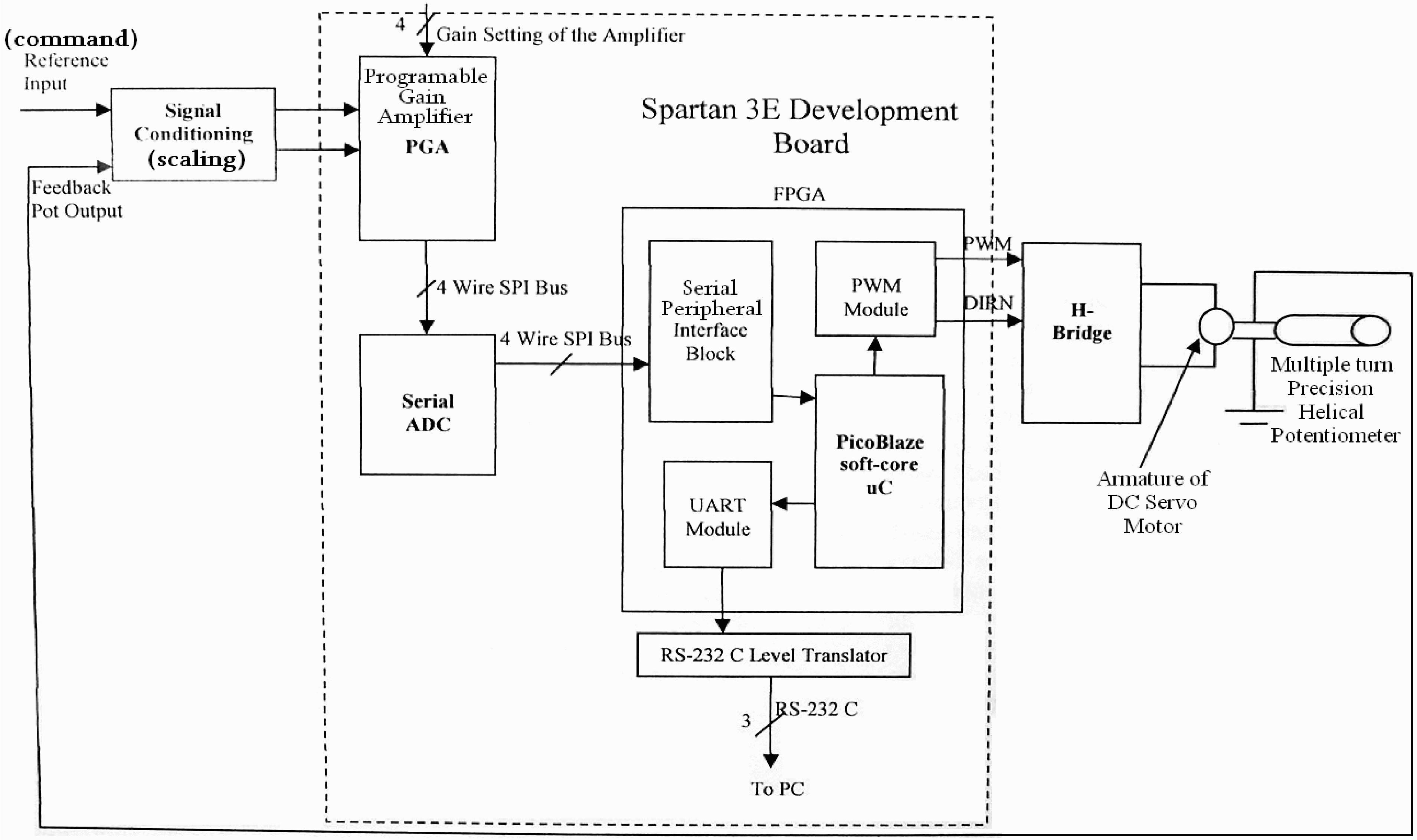
FPGA PID Controller Block Diagram


Input
Signal conditioning hardware
FPGA development board
A-to-D, PID design, digital signal out
H-bridge motor driver circuit
Motor
Sensor
Error
Sensor feedback

General steps for fast prototyping:

1. Derive basic mathematical model of motor & feedback
2. Identify constraints for hardware & software
3. Identify input and output signal constraints

System Functional Block Diagram



Positional Control System Functional Block Diagram

9-step Prototyping Procedure

1. Motor armature modeling
2. MATLAB: Find P,I, & D terms and time contraints
3. Xilinx soft core assembly language for PID design

4. Code the interfaces for the VHDL modules
5. Make connections schematically, download to FPGA
6. Design electronic interfaces to FPGA board

7. Test the control loop system
8. Tune the PID design based on the test
9. Design a LabView module to display system response