Physics 124: Lecture 9

Project-related Issues

Analog Handling

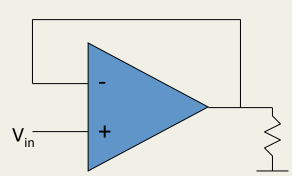
- Once the microcontroller is managed, it's often the analog end that rears its head
  - getting adequate current/drive
  - signal conditioning
  - noise/glitch avoidance
    - debounce is one example
  - dealing with crude simplicity of analog sensors

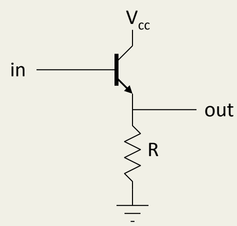
Computers are pretty dumb

- Operating in the real world requires advanced pattern recognition
  - the Achilles Heel of computers
  - examples of failures/disappointments
    - voice recognition (simple 1-D time series, and even *that's* hard)
    - autopilot cars?
    - intolerance for tiny mistakes/variations
  - many projects require discerning where a source is, avoiding obstacles, ignoring backgrounds, etc.
    - just keep in mind that things that are easy for our big brains (which excel at pattern matching; not so good at tedious precision) may prove very difficult indeed for basic sensors and basic code

Getting Enough Current

- Some devices/sensors are not able to source or sink much current
  - Arduino can do 40 mA per pin, which is big for this business
- On the very low end, an op-amp buffer fixes many ills
  - consider phototransistor hooked to 3 kΩ sensing resistor
  - we're talking mA of current, so drawing even 0.5 mA away from the circuit to do something else will change the voltage across the resistor substantially
  - enter op-amp with inverting input jumped 'round to output
  - can now source something like 25 mA without taxing V_{in} one iota

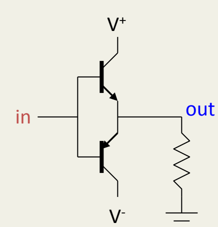




Transistor Buffer

- In the hookup above (emitter follower), $V_{out} = V_{in} - 0.6$
  - sounds useless, right?
  - there is no voltage "gain," but there *is* current gain
  - Imagine we wiggle V_{in} by ΔV: V_{out} wiggles by the same ΔV
  - so the transistor current changes by $\Delta I_e = \Delta V/R$
  - but the base current changes $1/\beta$ times this (much less)
  - so the "wiggler" *thinks* the load is $\Delta V/\Delta I_b = \beta \cdot \Delta V/\Delta I_e = \beta R$
  - the load therefore is less formidable
- The "buffer" is a way to drive a load without the driver feeling the pain (as much): it's impedance isolation

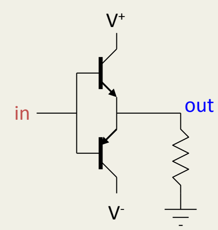
Push-Pull for Bipolar Signals



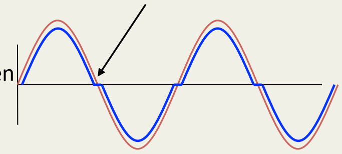
- Sometimes one-sided buffering is not adequate
  - need two transistors: npn for + side, pnp for –
  - idea is that input sees high-impedance
  - the current into the base is < 1/100 of I_{CE}
  - load current provided by power supply, not source
- Called a Push-Pull transistor arrangement
- Only problem is "crossover distortion"
  - npn does not turn on until input is +0.6 V
  - pnp does not turn on until input is < −0.6 V
  - creates dead-zone in between

Hiding Distortion

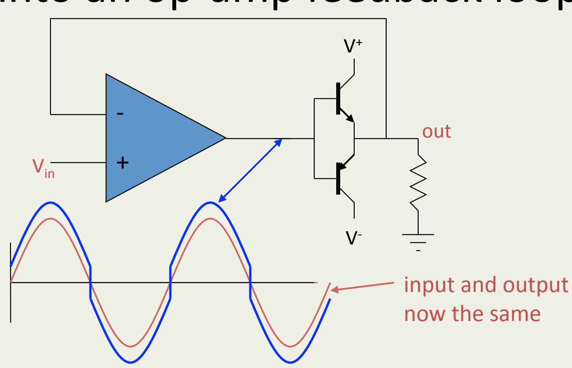
- Consider the "push-pull" transistor arrangement to the right
  - an npn transistor (top) and a pnp (bottom)
  - wimpy input can drive big load (speaker?)
  - base-emitter voltage differs by 0.6V in each transistor (emitter has arrow)
  - input has to be higher than ~0.6 V for the npn to become active
  - input has to be lower than -0.6 V for the pnp to be active



- There is a no-man's land in between where neither transistor conducts, so one would get "crossover distortion"
  - output is zero while input signal is between -0.6 and 0.6 V

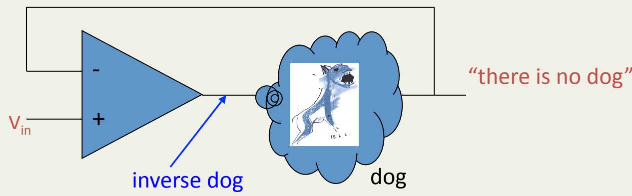


Stick it into an op-amp feedback loop!



- By sticking the push-pull into an op-amp's feedback loop, we guarantee that the output faithfully follows the input!
  - after all, the golden rule for op-amps demands that + input = - input
- Op-amp jerks up to 0.6 and down to -0.6 at the crossover
  - it's almost magic: it figures out the vagaries/nonlinearities of the thing in the loop
- Now get advantages of push-pull drive capability, without the mess

Dogs in the Feedback

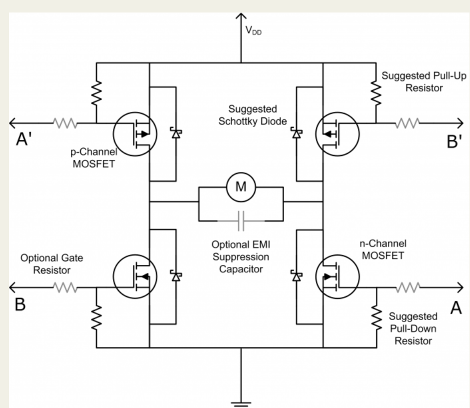


- The op-amp is obligated to contrive the inverse dog so that the ultimate output may be as tidy as the input.
- Lesson: you can hide nasty nonlinearities in the feedback loop and the op-amp will "do the right thing"

We owe thanks to Hayes & Horowitz, p. 173 of the student manual companion to the *Art of Electronics* for this priceless metaphor.

MOSFETs often a good choice

- MOSFETs are basically voltage-controlled switches
  - n-channel becomes "short" when logic high applied
  - p-channel becomes "short" when logic low applied
  - otherwise open
- Can arrange in H-bridge (or use pre-packaged H-bridge on a chip)
  - so A=HI; A'=LOW applies VDD to left, ground to right
  - B=HI; B'=LOW does the opp.
  - A and A' always opposite, etc.
  - A and B default to LOW state



Timing Issues

- Microcontrollers are fast, but speed limitations may well become an issue for some
- Arduino processor runs at clock speed of 16 MHz
  - 62.5 ns = 0.0625 μs
  - machine commands take 1, 2, 3, or 4 cycles to complete
    - see chapter 32 of datasheet (pp. 537–539) for table by command
  - but Arduino C commands may have dozens of associated machine commands
    - for example, `digitalWrite()` has 78 commands, though not all will be visited, as some are conditionally branched around (~36 if not PWM pin)
    - testing reveals 4 μs per `digitalWrite()` operation (5 if PWM pin)
    - implies about 64 (80) clock cycles to carry out

Timing Exploration, continued

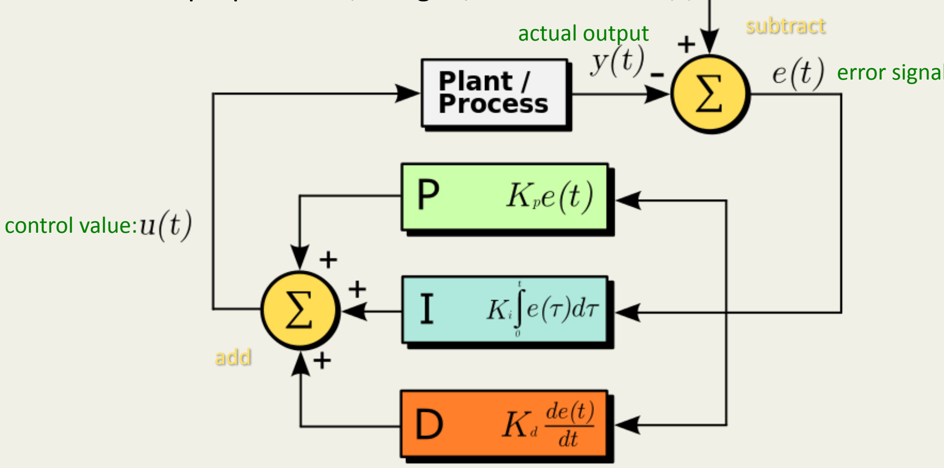
- Program is basically repetitive commands, with `micros()` bracketing actions
  - `micros()` itself (in 16 repeated calls, nothing between) comes in at taking 4 μs to complete
  - `Serial.print()` takes 1040 μs per character at 9600 baud
    - 8 data bits, start bit, stop bit → 10 bits, expect 1041.7 μs
    - `println()` adds 2-character delay
  - `digitalRead()` takes 4 μs per read
  - `analogRead()` takes 122 μs per read
- Also keep in mind 20 ms period on servo 50 Hz PWM
- And when thinking about timing, consider inertia
  - might detect obstacle 5 cm ahead in < 1 ms, but can you *stop* in time?

Another Way to Explore Timing

- Don't be shy to use the oscilloscope
  - a pair of `digitalWrite()` commands, HIGH, then LOW, will create a pulse that can be easily triggered, captured, and measured
  - for that matter, you can use digital output pins expressly for the purpose of establishing relative timings between events
  - helps if you have to choreograph, synchronize, or just troubleshoot in the time domain
  - think of the scope as another debugging tool, complementary to Serial, and capable of faster information

Control Problems

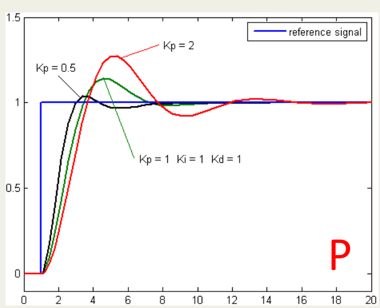
- When it comes to controlling something through feedback, always think PID first
  - PID: proportional, integral, derivative

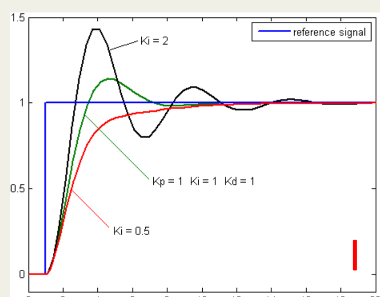


PID, in pieces

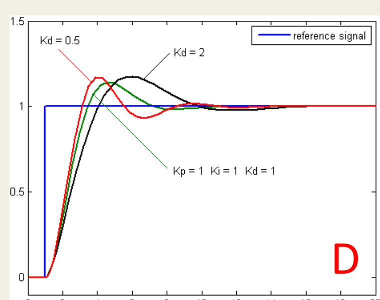
- Proportional (Ghost of Conditions Present)
  - *where are we now?*
  - simple concept: take larger action for larger error
  - in light-tracker, drive more degrees the larger the difference between phototransistors
  - higher gain could make unstable; lower gain sluggish
- Integral (Ghost of Conditions Past)
  - *where have we been?*
  - sort of an averaging effect: error × time
  - responds to nagging offset, fixing longstanding errors
  - looking to past can lead to overshoot, however, if gain is too high
- Derivative (Ghost of Conditions Future)
  - *where are we heading?*
  - damps changes that are too fast; helps control overshoot
  - gain too high amplifies noise and can produce instability

PID, in pictures





- Impact of changing different gains, while others held fixed
  - blue is desired response
  - green is nominal case
    - $K_p = K_i = K_d = 1$ in this case
  - ideal values depend on system



Tuning PID Control

- See http://en.wikipedia.org/wiki/PID_controller
- One attractive suggested procedure:
  - first control the system only with proportional gain
  - note ultimate gain, K_u, at which oscillation sets in
  - note period of oscillation at this ultimate gain, P_u
  - If dealing with P only, set $K_p = 0.5K_u$
  - If PI control: set $K_p = 0.45\ K_u$; $K_i = 1.2K_p/P_u$
  - If full PID: $K_p = 0.6K_u$; $K_i = 2K_p/P_u$; $K_d = K_p \times P_u/8$
- Control Theory is a rich, complicated, PhD-earning subject
  - not likely to *master* it in this class, but might well scratch the surface and use some well-proven techniques

Announcements

- Project Proposals due Friday, 2/5, in class
- Week 5 lab:
  - could work on light-tracker (due by next week, 2/9, 2/10)
  - could work on proposals with "consultants" at hand
    - due at week's end
- Following week we'll begin project mode, with new schedule
- Please fill out mid-quarter evaluation by Monday:
  - https://academicaffairs.ucsd.edu/Modules/Evals