Lecture 25

Debugging Noise and Other Problems

Agenda

- Noise sources and how to deal with them
- Encoders
- Example Debugging DIY Arduino Code

Design Review in Recitation

- Contents of presentation
  1. Overall concept: abilities you want your robot to achieve
  2. Port allocation (which pins will do what function) and CPU architecture – e.g. a block diagram
  3. General software approach
  4. Mechanical design overview (include any particularly hard issues)
  5. Should have most purchase done by now – include list of purchases in submission (don't need to review this)

Grading Design Review

- Primary grade based on teaching staff interpretation of your likelihood of success in the competition.
  - Mostly depends on your progress to date.
  - Also your designs and plan.

01

Noise and how to deal with it

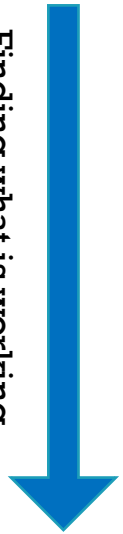
What is noise?

- How does noise effect our circuits?
- Has anyone had noise issues ?
- Noise can come into sensor readings, but can they affect digital logic?
- How can you get rid of noise?
- How does noise get in to your circuits??

Standard Debugging process

1. Start with most reliable:
    1. Check power and ground, trace power to and ground to your system.
2. Hardware components
    1. Check each hardware component output individually.
    2. Check inputs have proper effect (manually control input lines).
3. Software components
    1. Write short routines that you know work to check individual functions
4. Combine individual components one at a time until you isolate failure.
5. When everything is working separately except sometimes when assembled, **suspect noise as the problem**

Order of trust (correct and working properly)



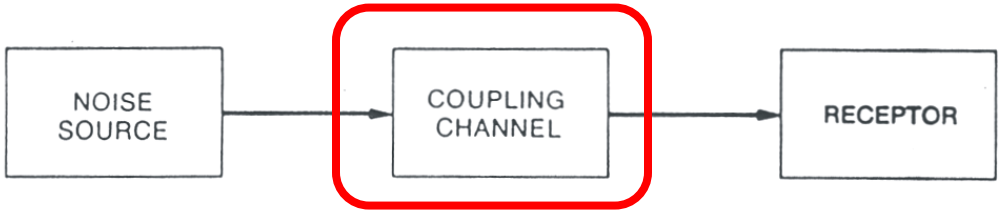
1. Corporate documentation
2. Test equipment:
   - E.g. DMM, Power Supply, Oscilloscope
3. Passive parts
   - Resistors, caps, wires
4. Simple active parts
   - LEDs, diodes, voltage regulators
5. Oscillosorta
6. Lecture documentation
7. Web documentation

90% of ERRORS DUE TO THESE:

12. Blown port on MCU
13. External hardware
    - H-bridge, logic, opamps, sensor, motor, driver
14. Wiring
    - breadboard interface, connectors,
15. Software

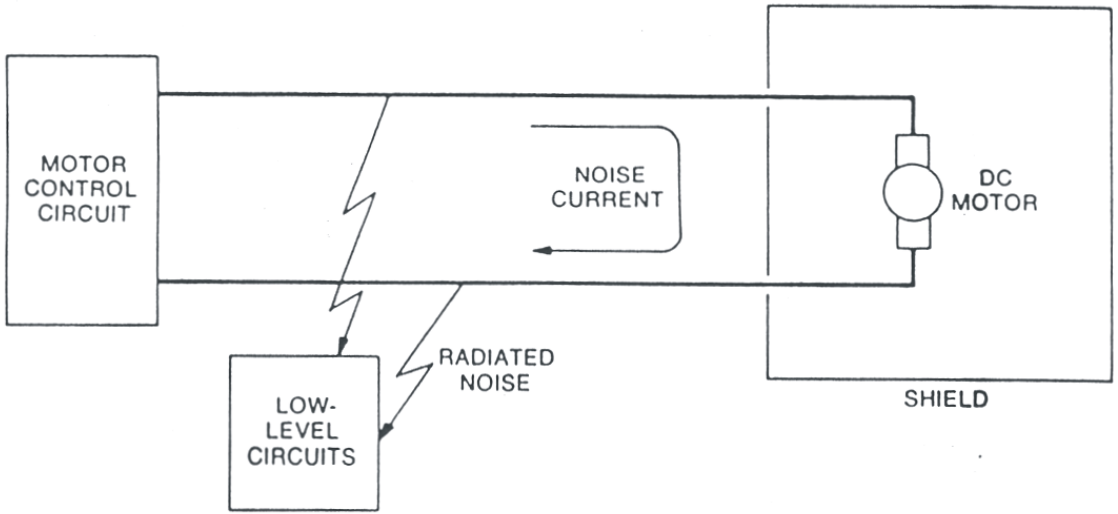
Intermittent failure. It's either interrupts or NOISE...

How noise gets into your circuits



Four types of coupling

1. Conductive coupling
2. Capacitive coupling
3. Inductive coupling
4. Radiative coupling



Characteristics of the noise sources

- Distance from the victim

Most common, easiest to fix

Distance = 0 → Direct contact → Conductive coupling

RF noise

Distance > wavelength → Probably Radiative 80Mhz = ~4m

0 < Distance < Wavelength → Capacitive or Inductive

- Voltage noise source

Any two conductors with shared area

Strong electric field → Capacitive coupling

'High' dV/dt → Conductive and Capacitive coupling

Examples?

Parallel wires

Digital signals!

- Current noise source

'High' current → Strong magnetic field → Inductive coupling

$V = L\, dI/dT$

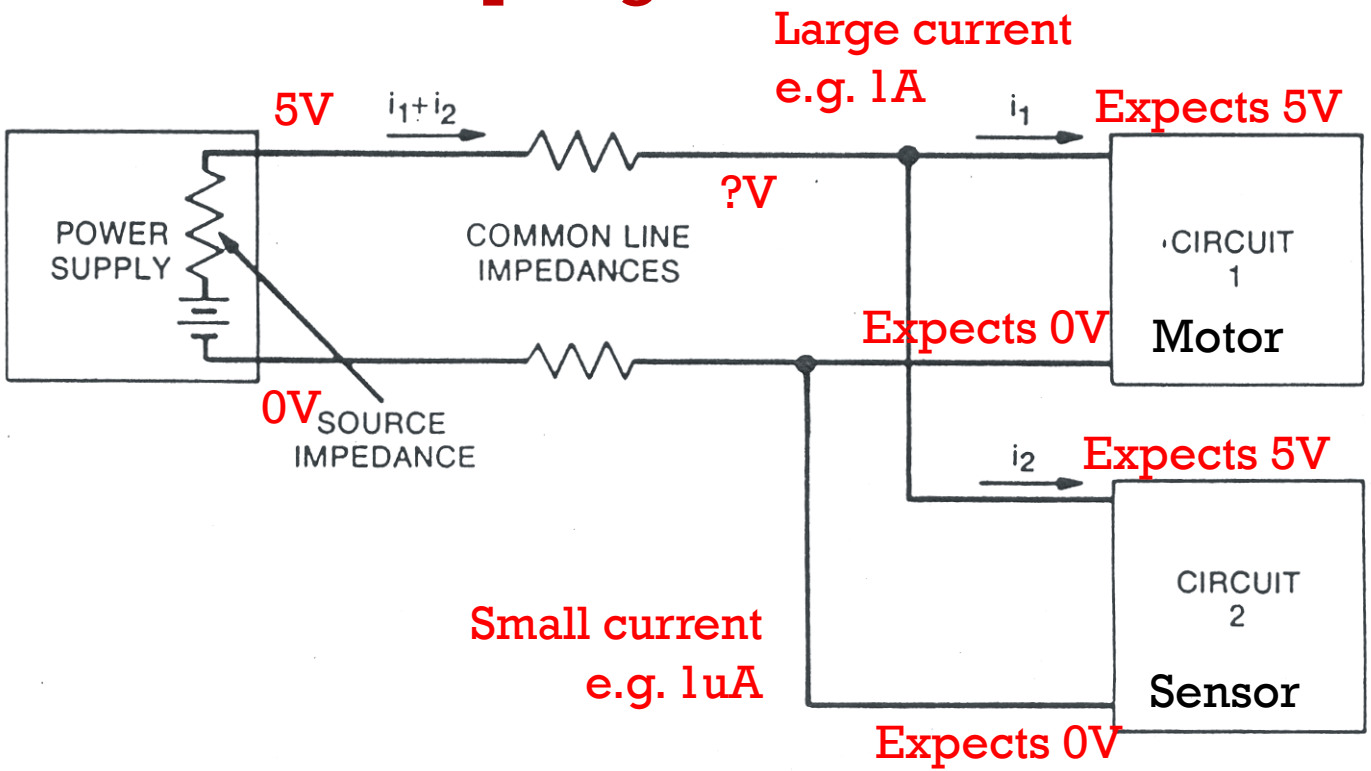
Examples?

Motors

- Frequency noise source

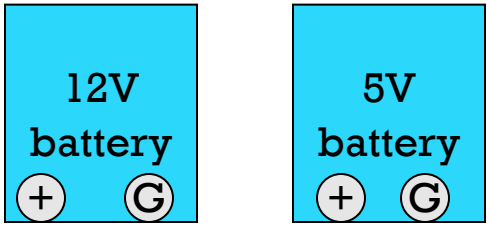
'High' frequency → Radiation → Radiative and inductive coupling

Conductive Coupling

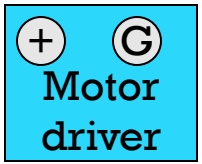


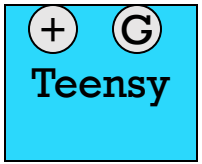
Q1: If cabling has 0.5 resistance, what voltage will the sensor see at its power?

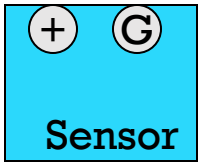
Q2: Draw power and ground sources to the devices



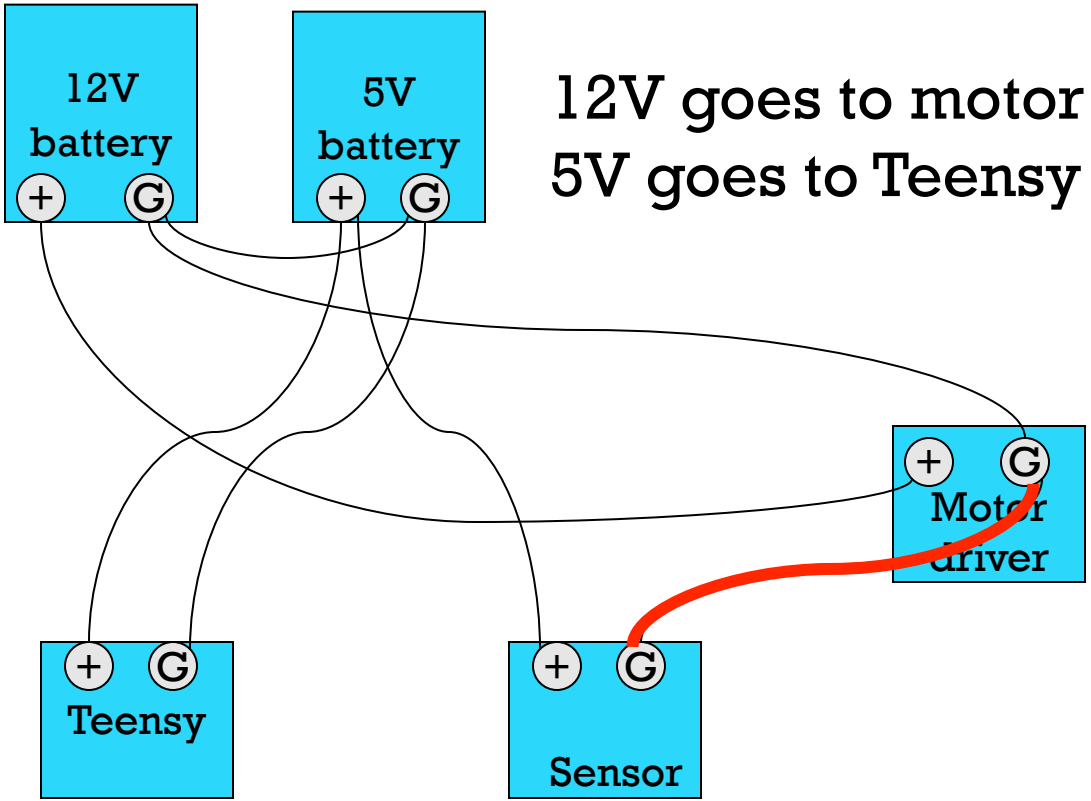
12V goes to motor driver
5V goes to Teensy and sensor





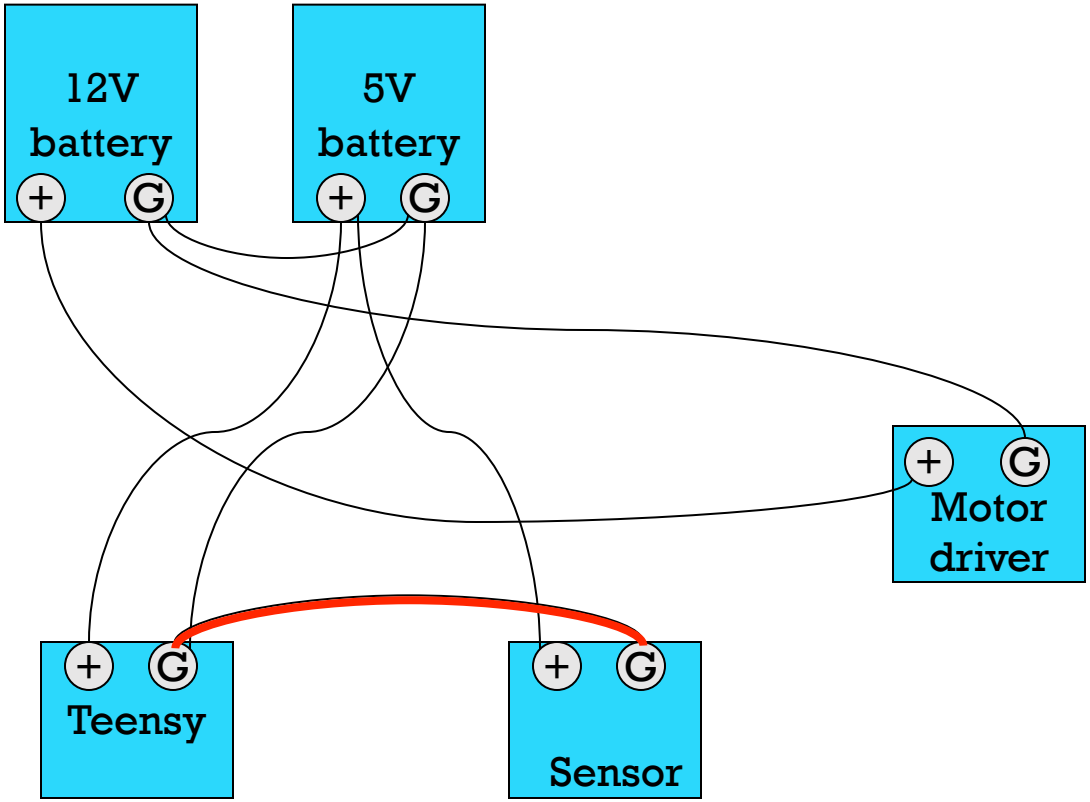


Q3: In chat what's wrong with this?

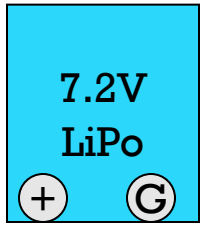


12V goes to motor driver
5V goes to Teensy and sensor

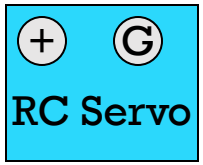
How about this?

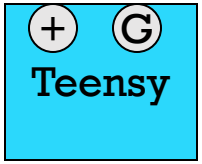


If we use a LiPo, how do we get 5V?



Voltage Regulation?

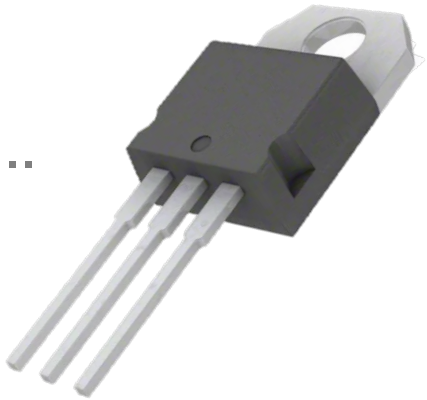




Requires 5V @ ~200mA max
No onboard regulator

Most servos will take
4V to 6V
Max current varies
Often 1A – 3A

Voltage regulators – LM7805

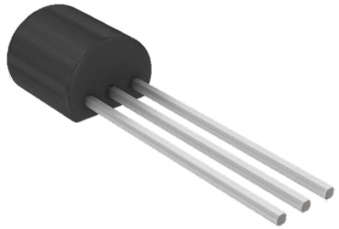


- In general – it's an old standard, low-cost, reliable…
- 5V Fixed regulator
  - 1.5A max current output
  - 35V max input
  - 7.5V minimum
- LM7805: … *employ internal current-limiting, thermal shutdown and safe-area compensation, making them essentially indestructible. If adequate heat sinking is provided, they can deliver over 1.5-A output current.*

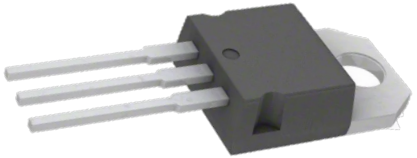
- Heat generation is a function of both current output as well as input voltage.

Other GMLab options for generating ~5V

- MCP1702 LDO regulator
  - Vin_{max}=13.2V, Vin_{min} =Vout + 0.62V
  - Vout = 5V (also have 3.3V version)
  - 250mA max output



- LM317 Adjustable Regulator,
  - Vin_{min} = Vout + 3V
  - 1.5A max output

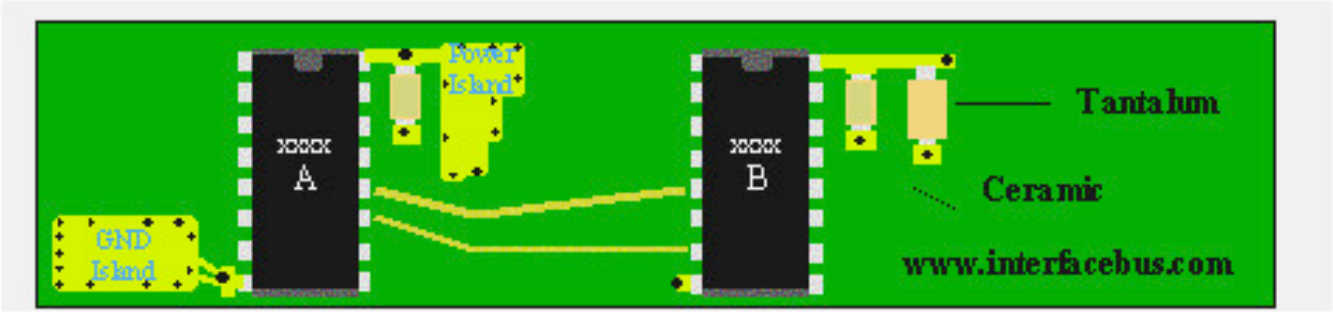


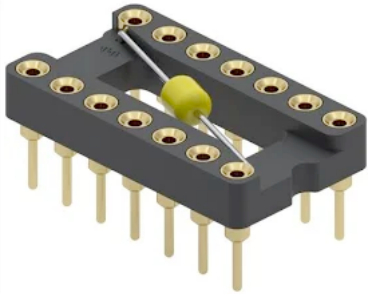
- TCA0372 Dual Power Op-Amp,
  - Vin_{min} ~= Vout + 1V
  - Won't be fixed Vout
  - 1A max output

- DF005M Bridge Rectifier, 3W max
  - Vf = ~ 1V

Digital Circuitry Noise

- All digital circuits add noise potentially on the power lines.
- Add decoupling capacitors as standard practice
  - 10nF monolithic in parallel with 10uF tantalum or electrolytic cap at power rails
  - 10nF monolithic at every digital chip as close as possible to power-gnd





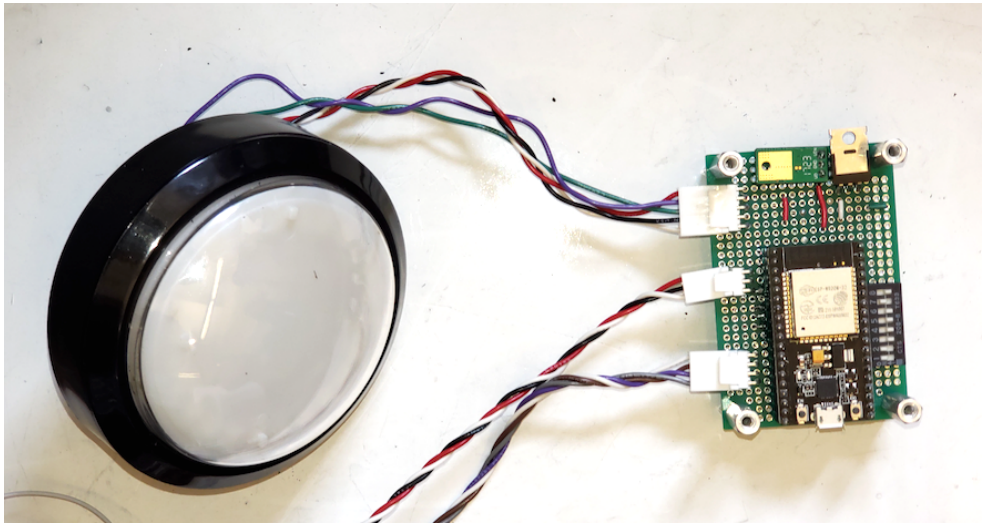
Reducing conductive coupling

- Change the conductive paths (e.g. ground lines)
- Reduce common impedance (star configuration connections all join at battery).
- Use separate power supplies (noisy vs clean) if possible.
- Use lower resistance wires (fatter) for high currents.

Low freq noise is hard to get rid of

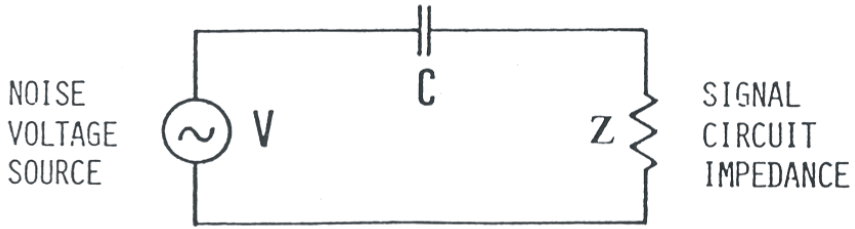
Capacitively coupled noise

Simplified circuit representing all types of capacitive coupling



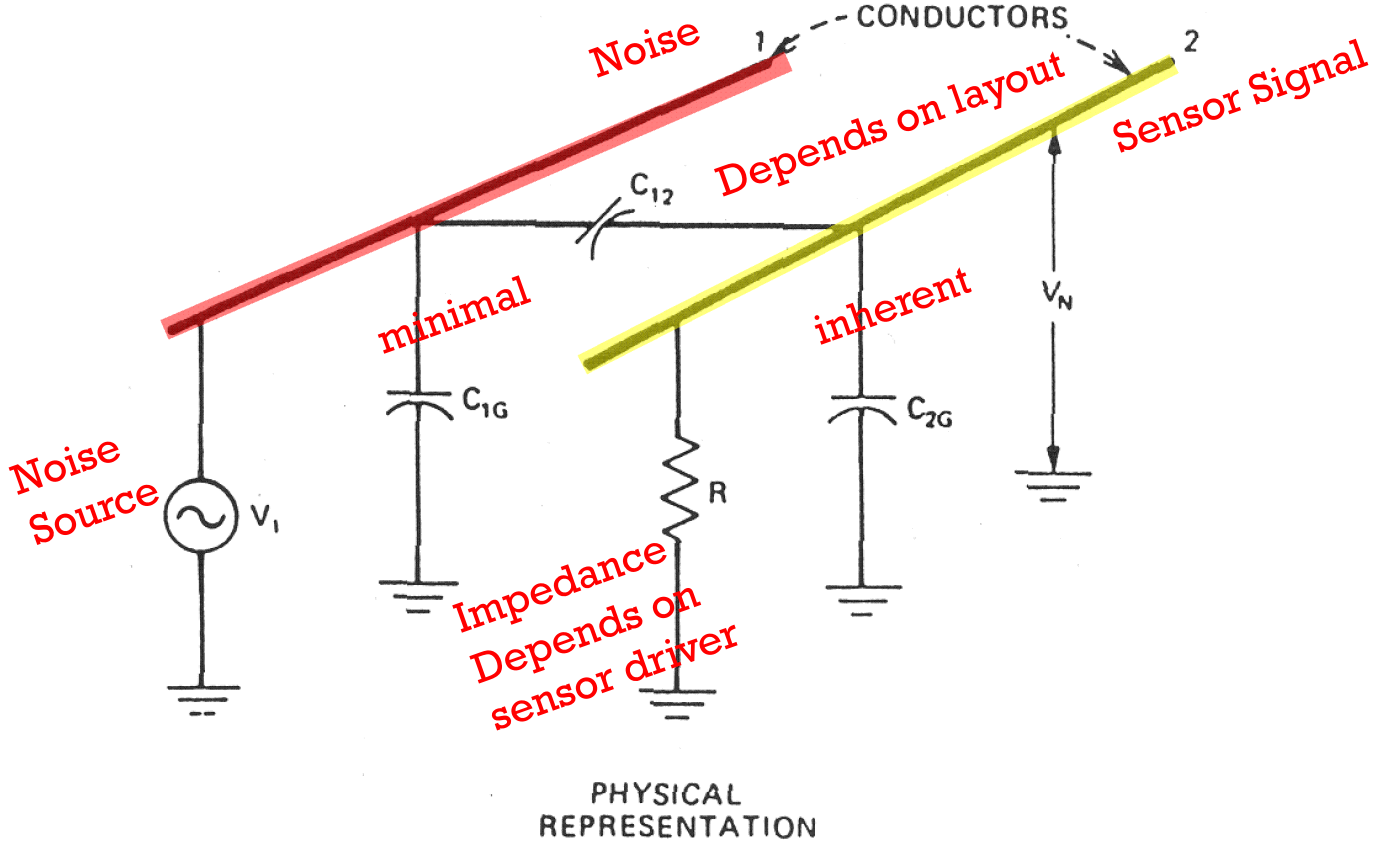
Coupling capacitance

Shared area

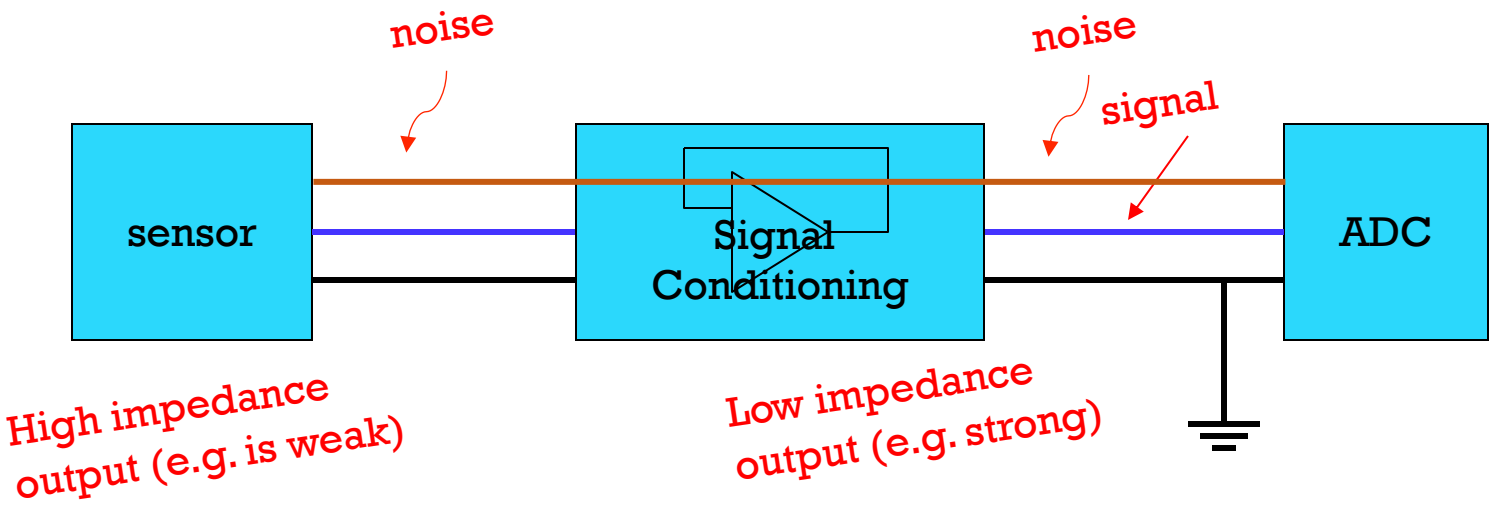


Typical example is leads next to each other, or long parallel cables

Physical Representation of Capacitively Coupled Noise

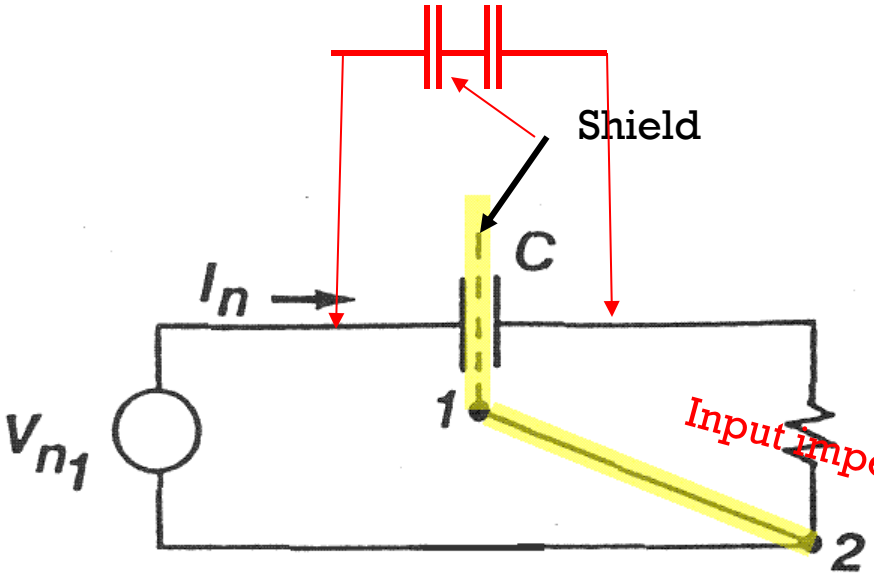


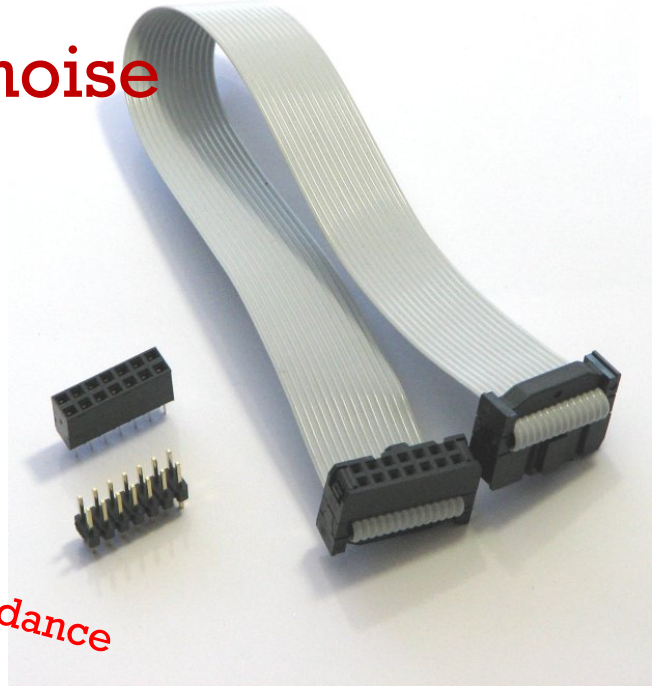
Sensor wiring



Want lower impedance at ADC input means higher currents at signal conditioning output
(e.g. op amp output driving larger current)

Reducing Capacitively coupled noise





Position the shield to intercept the noise current

Connect the shield to return the noise current to the source

Summary of capacitive noise reduction techniques

Reduce capacitive noise coupling by

1. Reducing coupling capacitance
2. Reducing circuit impedance
3. Using shielding

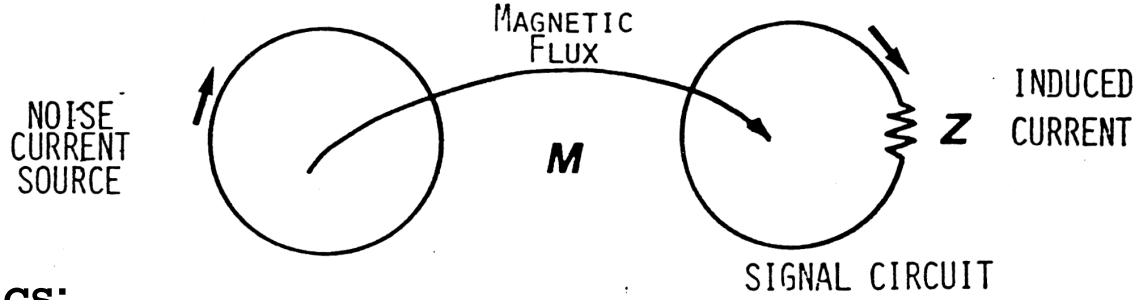
e.g. reduce parallel lines

e.g. increase op amp drive

Capacitive shielding requires

1. Proper shield location
2. Correct shield connection

Inductive Noise Coupling



Characteristics:

- Large loop areas in wiring
- High noise current or frequency
- Unaffected by non-conducting, non-magnetic materials
- Shield effectiveness not changed by grounding
- Detectable by magnetic (loop) sensor

Reducing Inductive Noise Coupling

Reduce the current in the noise source

Reduce the frequency content in the noise source

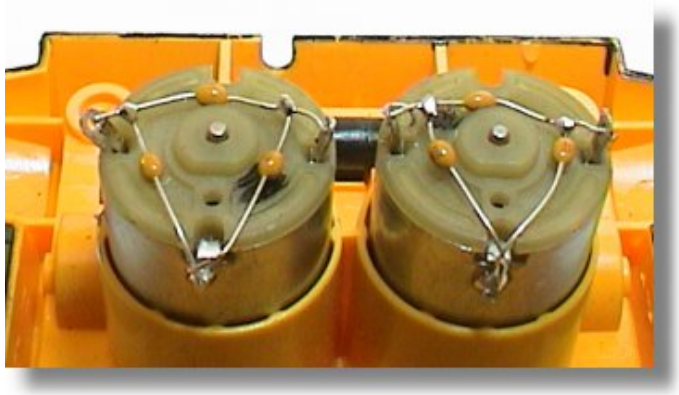
Reduce the mutual inductance — e.g. makes loops smaller, or Increase distance

Use magnetic shielding — e.g. iron-based metal foil

Filter the noise at the signal circuit

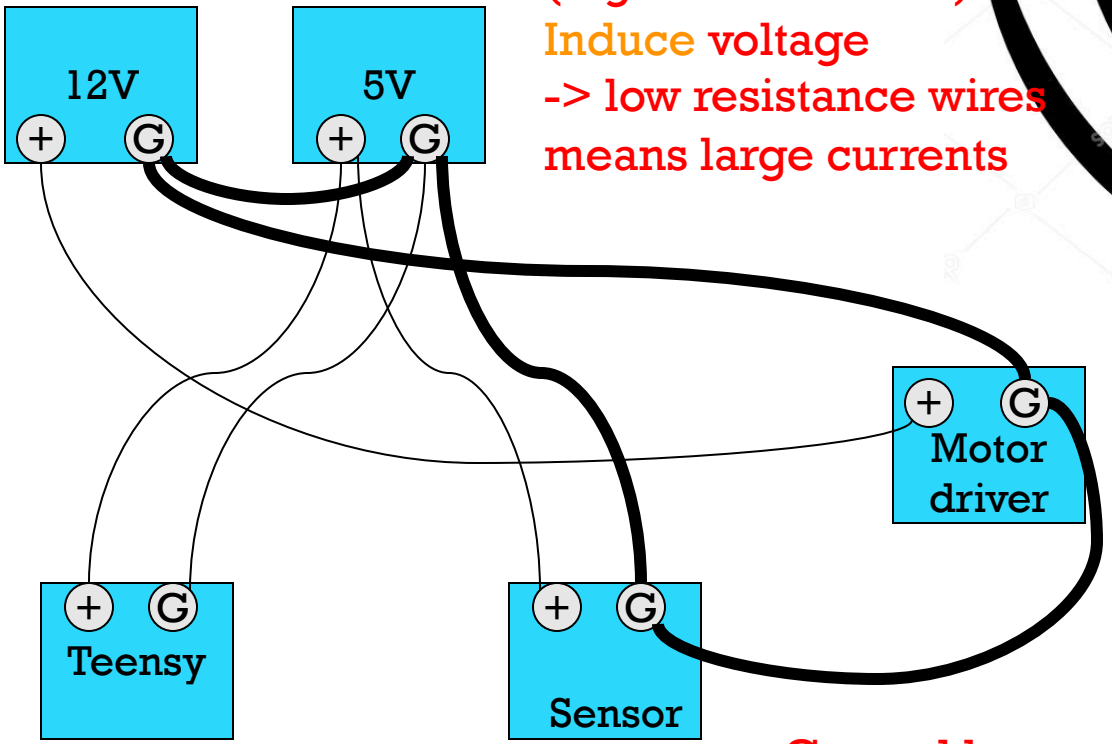
For motors, typically solder capacitor across leads and to motor casing.

Choose monolithic caps rated at least 3x's motor voltage



Loops in power

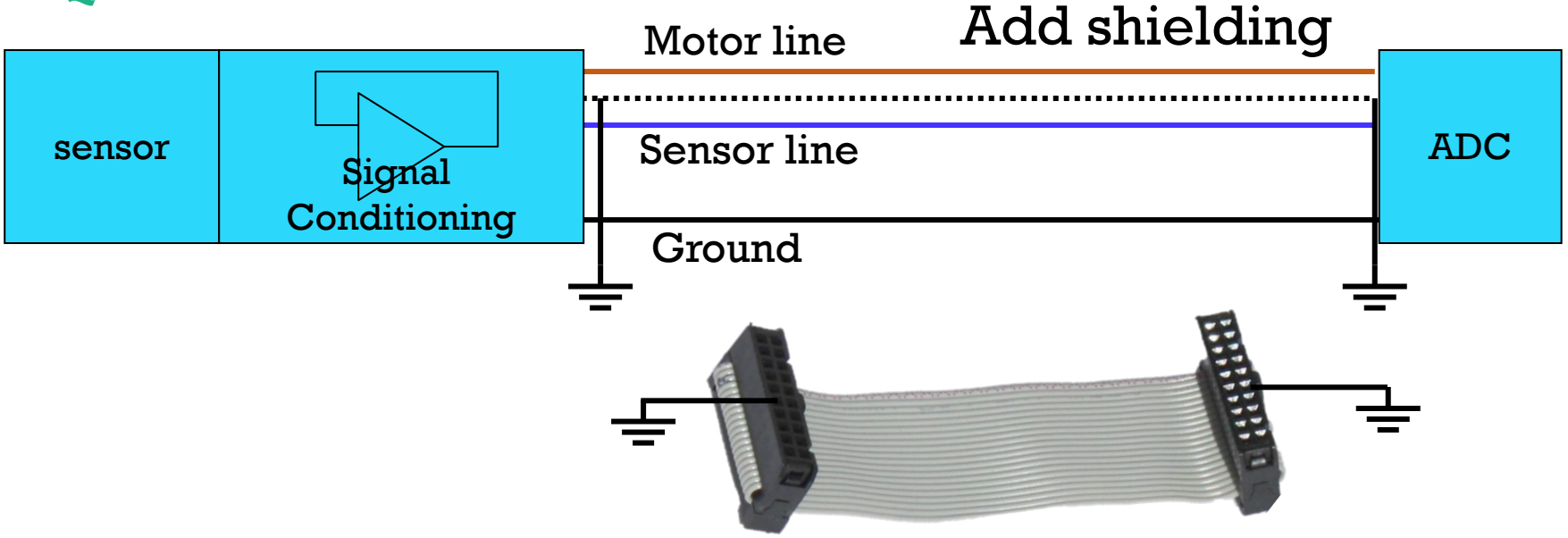
External fields
(e.g. 60Hz ambient)
Induce voltage
-> low resistance wires
means large currents



Ground loops are bad!

Q4: Where is there potential noise?

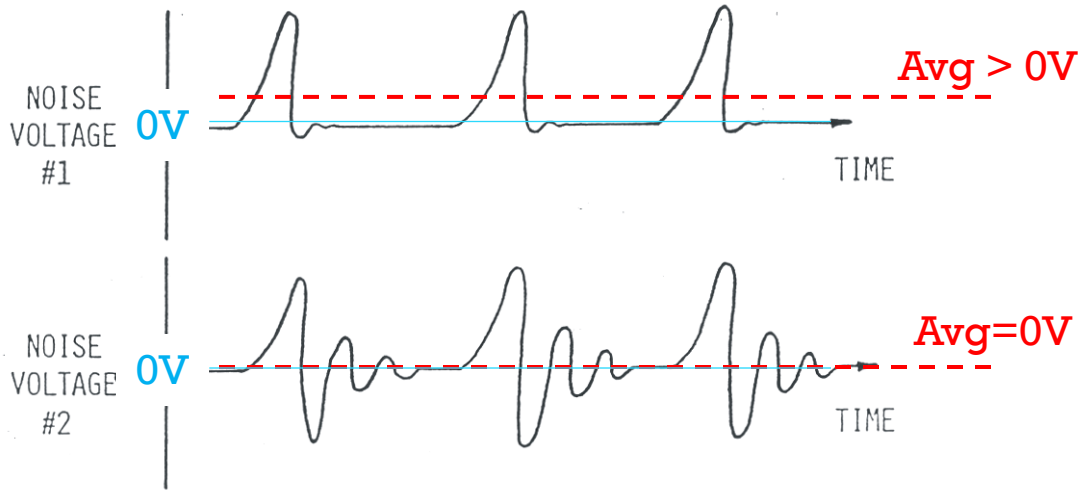
Q5: How about now?



Ground loops occur if you ground both ends of your shield

Q6: Which waveform must be from a conductively coupled source?

Pure signal is 0V

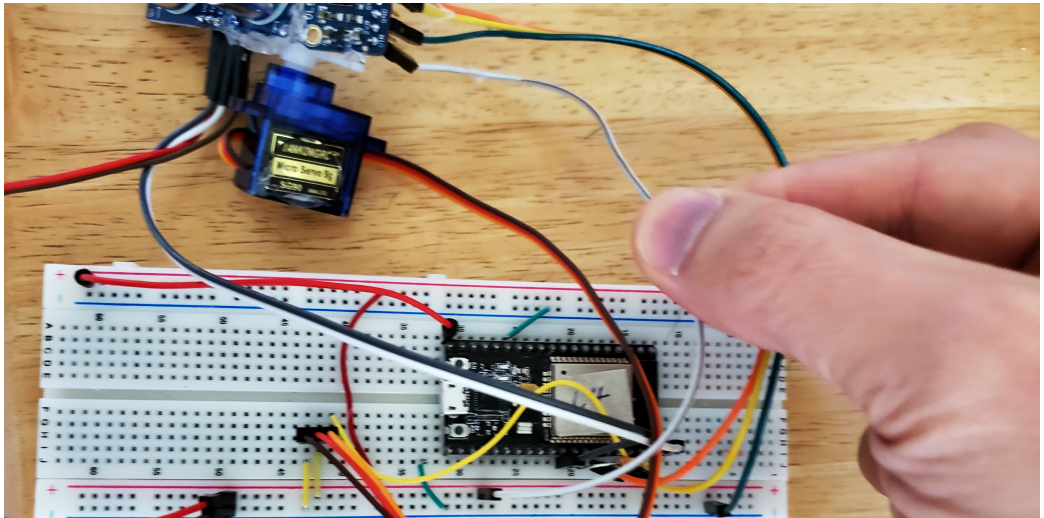


Why?

Identifying which type of coupling

- Non-zero average value for the waveform -> Conductive
- Conductive contact required -> Conductive
- Affected by people or cable movement -> Capacitive or Inductive

Wiggling wires shows some change in noise (not loose contact)



Isolating cross talk for ToF

- LED transmits at 10Mhz
- Summing phase shift gives time of flight for reflection
- Need to isolate photodiode from LED.

4 strategies to reduce crosstalk from LED to photodiode:

1. Separate ground planes surround each device
2. LED and photodiode leads are mounted perpendicular
3. Traces to diodes minimize loop area
4. Vias form vertical "picket fence"

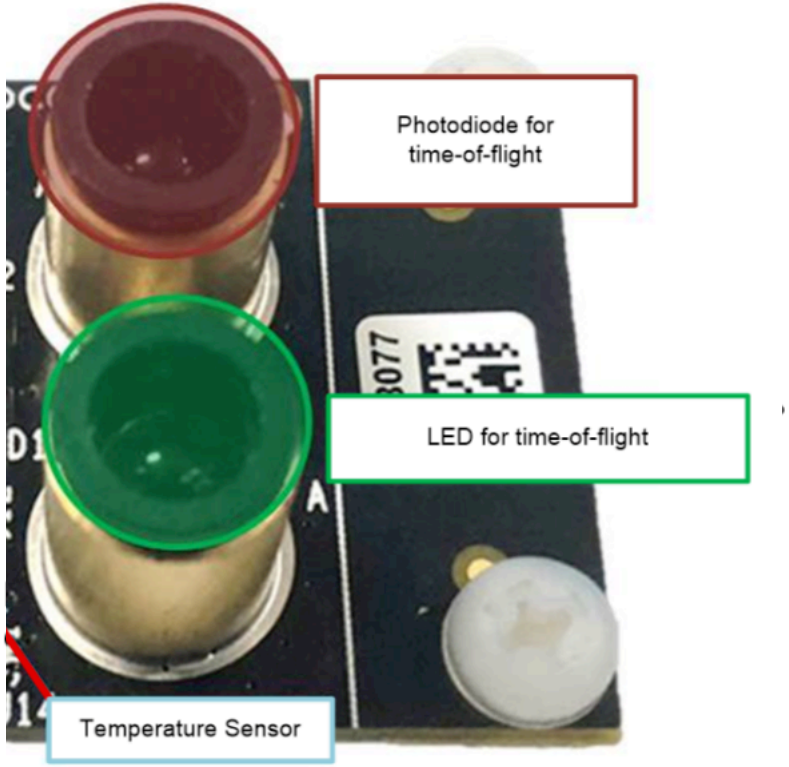
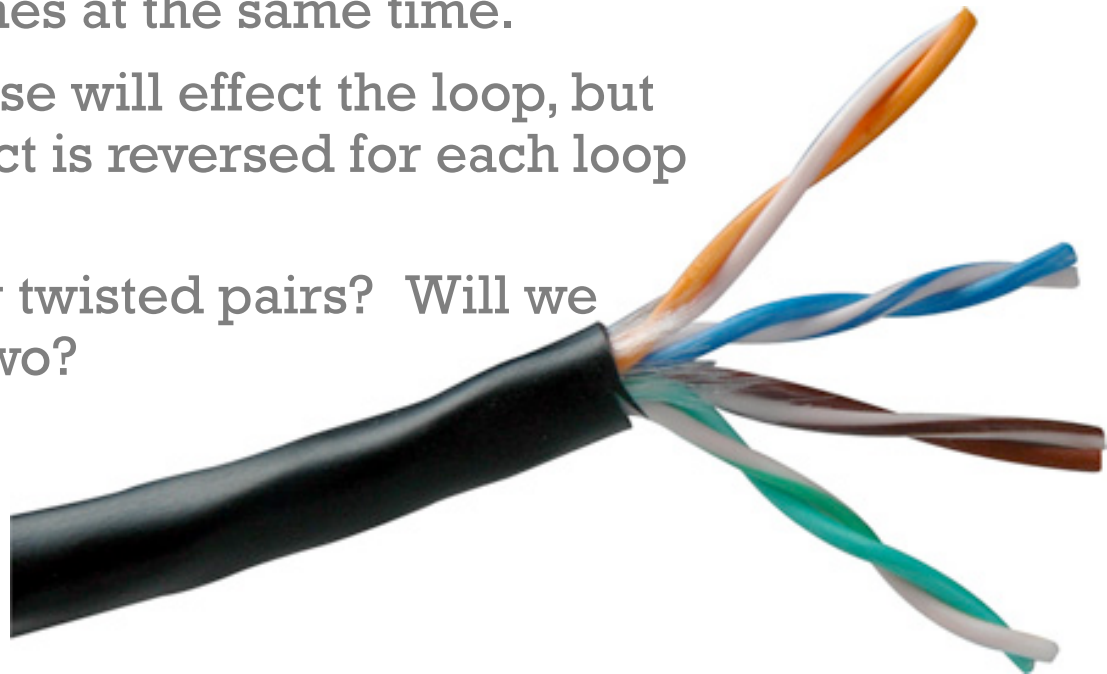


Figure 179. PCB Layout example

Q7: Cables often come in twisted pairs. Why? Especially considering capacitively coupled noise

- Twisted pairs are intended to be used differentially. Both lines hold one signal (so coupling is a good thing). Noise effects both lines at the same time.
- External fields inducing noise will effect the loop, but the twisting means that effect is reversed for each loop canceling out each other.
- But what about neighboring twisted pairs? Will we get crosstalk between the two?



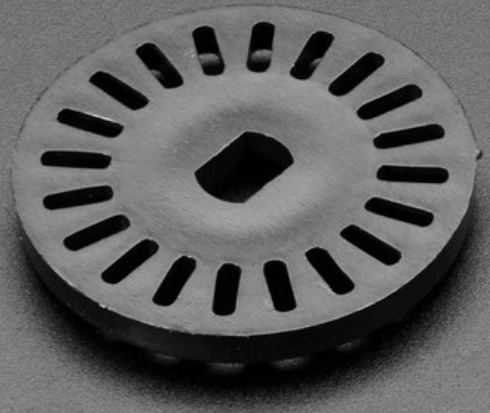
Noise Summary

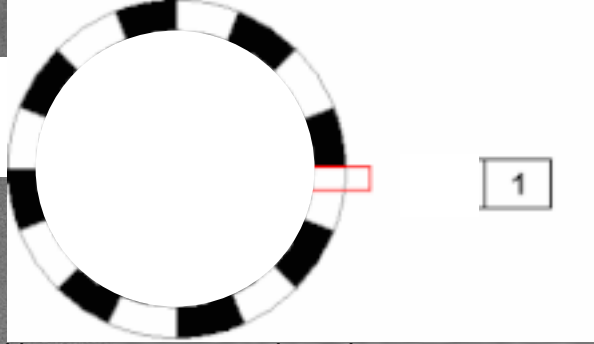
1. If needed, use two power supplies if you have noisy (e.g. motor circuits) and separate controllers
2. Add caps to motor leads
3. Add bypass caps
   A. each digital chips (.1uF monolithic)
   B. .1uF monolithic and ~10uF electrolytic @ power to board
4. Keep sensitive lines from high freq or high current lines (if must be close, cross at right angles)
5. Shield as necessary, attach only at one end!

02

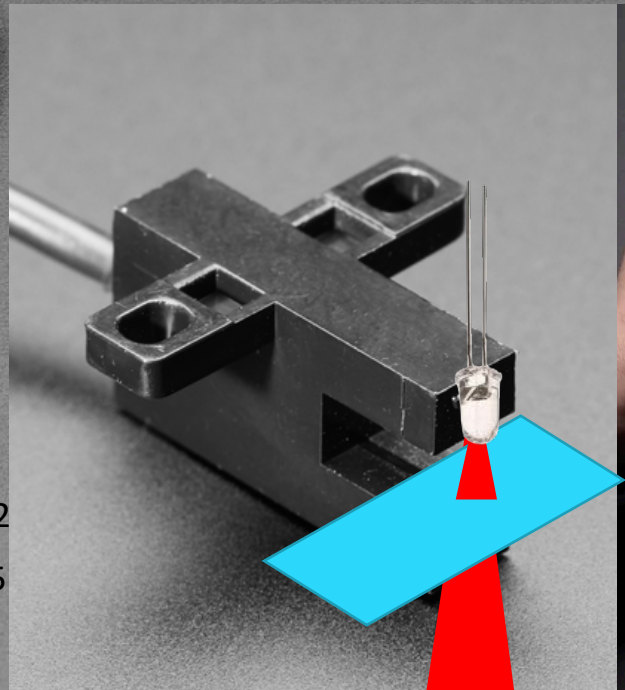
Rotary Position Encoders

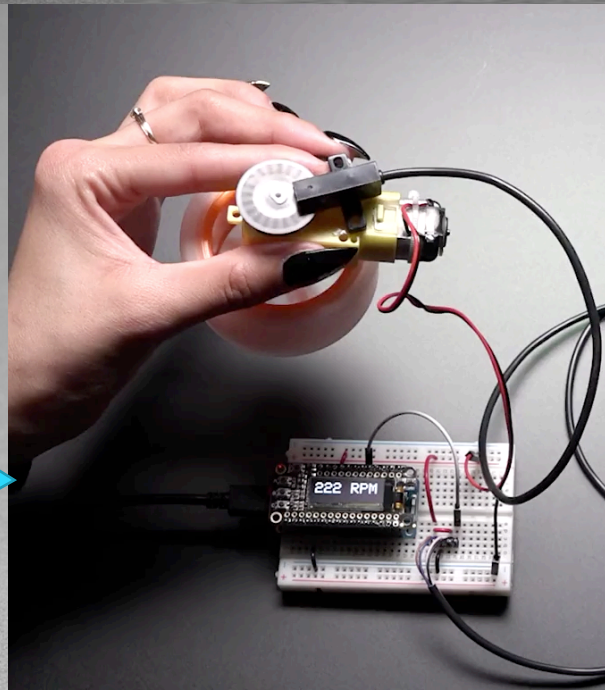
Optical Shaft Encoders





https://en.wikipedia.org/wiki/Rotary_encoder

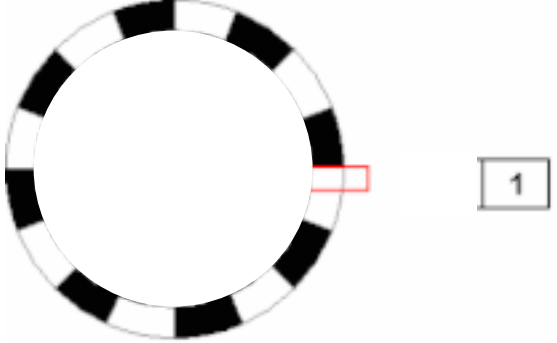




https://www.adafruit.com/product/3782

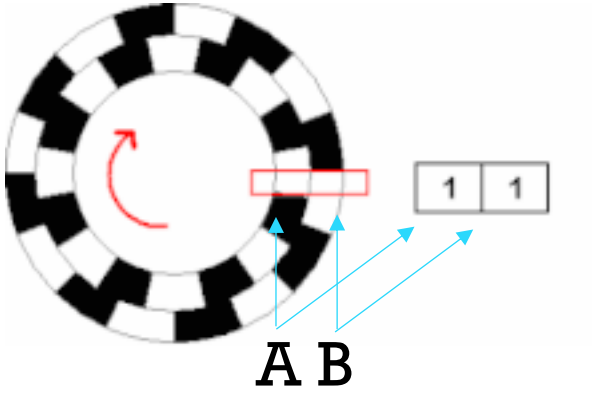
https://www.adafruit.com/product/3985

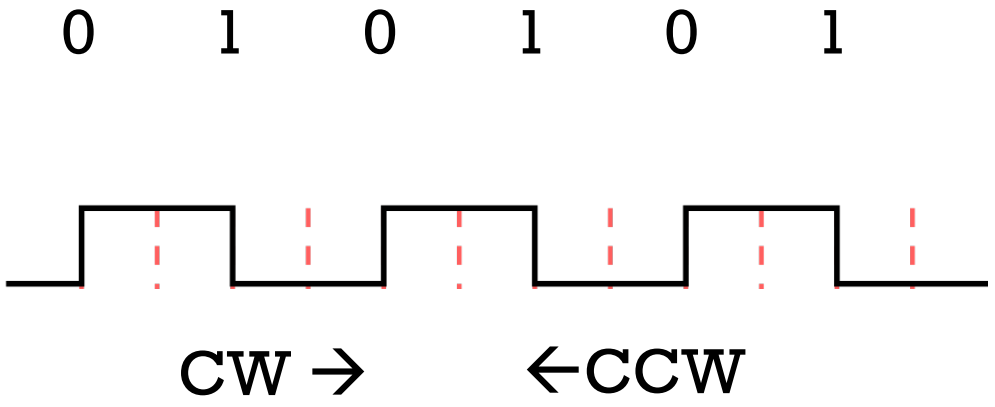
Quadrature



- With one square wave, we cannot tell what direction the wheel is spinning
- With two square waves 90o phase shifted, we can!

Q8: What is the sequence of logic level values for **A** for every rising edge of **B** as the time moves to the right?

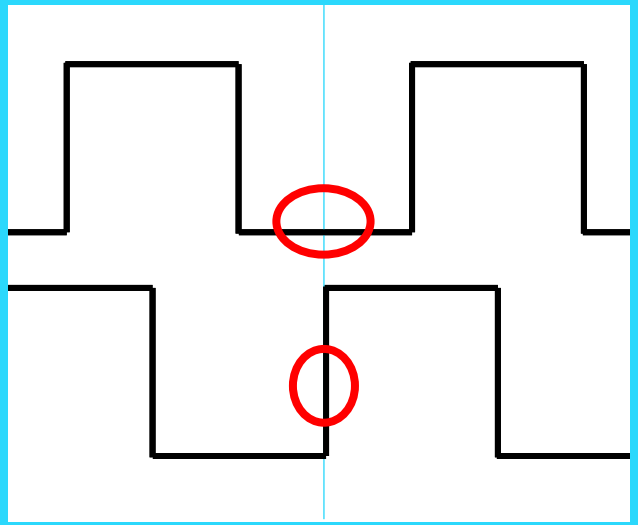




Quadrature intuition

Channel A

Channel B

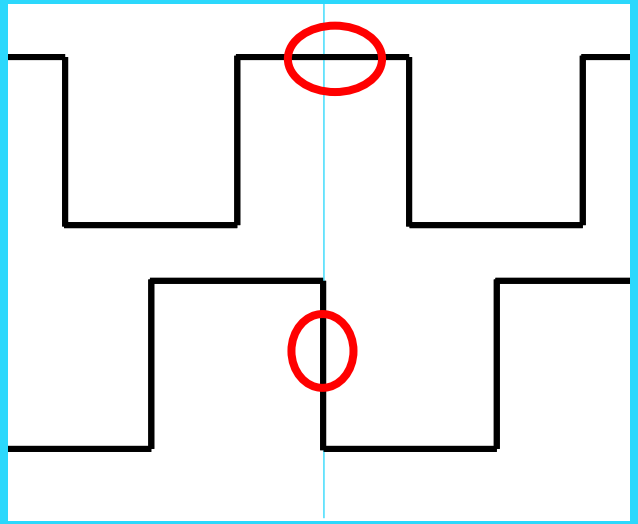


Channel A is low on rising edge of B when moving left

Quadrature intuition

Channel A

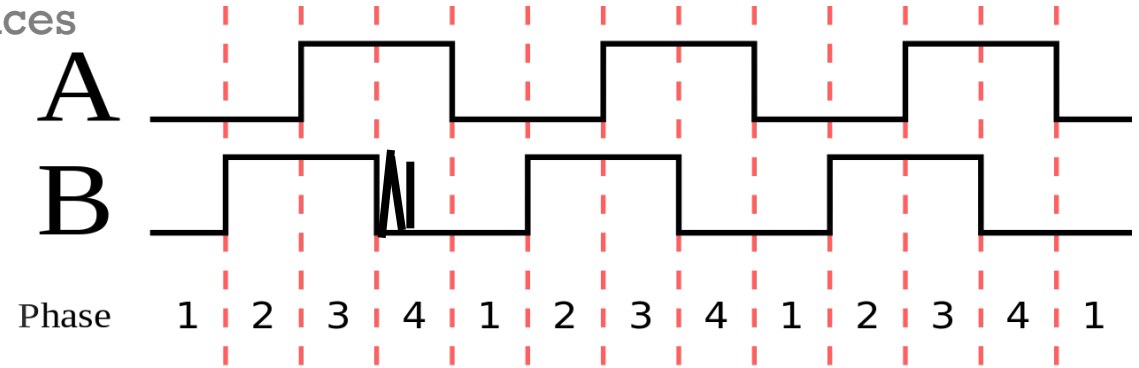
Channel B



Channel A is high on rising edge of B when moving right

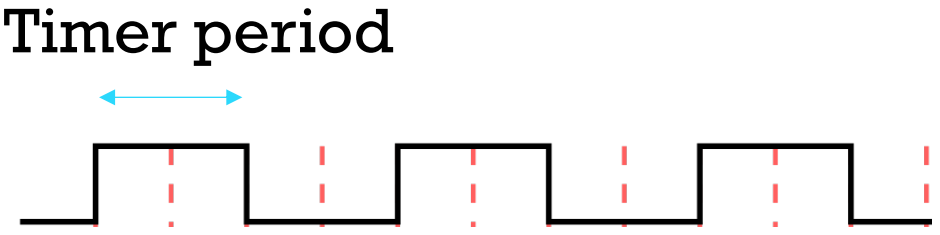
So what would code look like?

- Position Sensing – for incremental encoder
  - Periodically poll the state of A and B lines (need to do this faster than the highest expected frequency – e.g. 100RPM and resolution of 1024 = 102,400/60 = 1706 Hz). Polling digital lines can easily be done at >10kHz.
  - If there is a transition, then see which phase you are in. Can actually get 4x resolution by distinguishing between phases.
- Following phase transistions
  - More immune to bounces



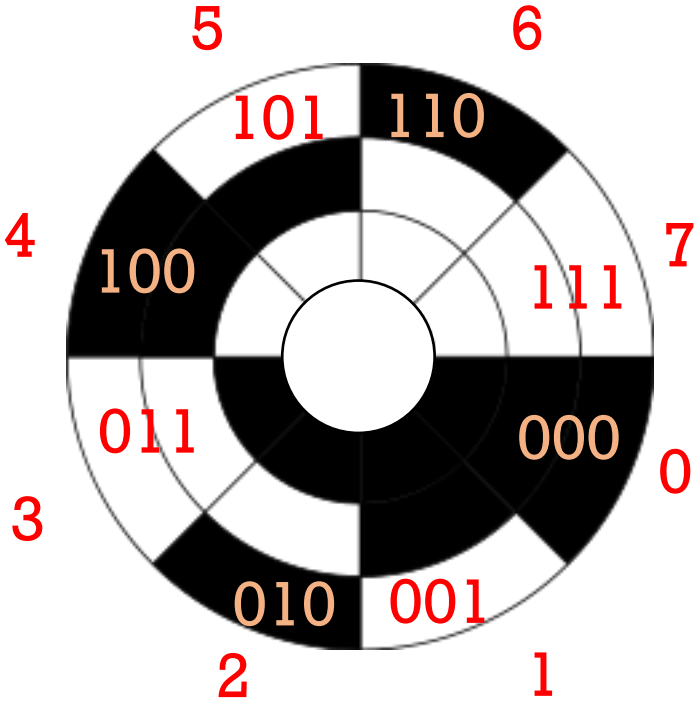
So what would code look like?

- Velocity Sensing – for incremental encoder
  - Can monitor position and difference based on timing.
  - Works well for fast velocities. Works poorly for slow velocities.
  - Instead use a timer to measure the pulse width. This can be done with interrupts to update a variable that stores the current velocity.
  - Need to handle the 0 velocity case as separate



Absolute Encoders

- Absolute encoders divide up the circle into unique codes for each sector.



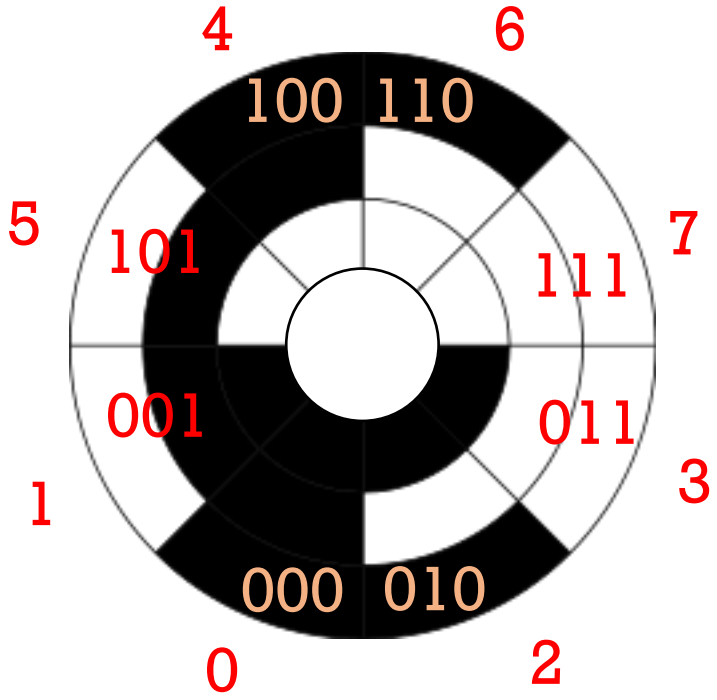
Absolue Encoder
Binary encoding

https://en.wikipedia.org/wiki/Rotary_encoder

Absolute Encoders

- Sectors are rearranged.

Q9 What is a feature of this pattern? What is useful about this pattern?



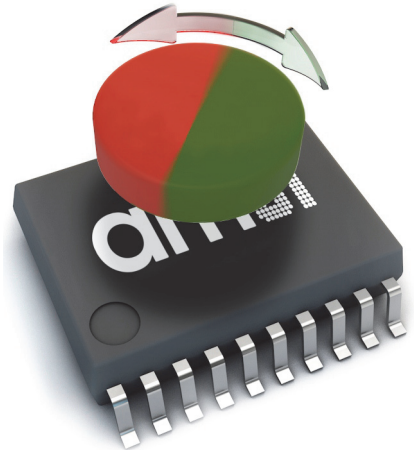
Absolue Encoder
Gray encoding

https://en.wikipedia.org/wiki/Rotary_encoder

Other rotary sensors

- Encoders
  - Absolute
  - Incremental
- Analog
  - Potentiometer
  - Synchro
  - Resolver

Mount magnet on shaft

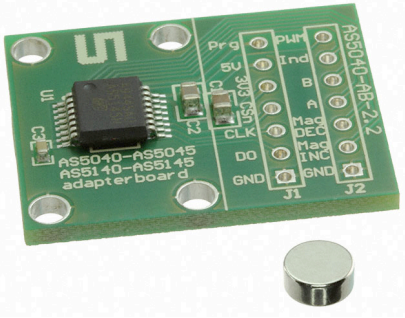


8.5 bit Magnetic encoders
AS5030-ATST ~$6 on digikey

Absolue Encoder
Gray encoding

https://en.wikipedia.org/wiki/Rotary_encoder

Example encoders



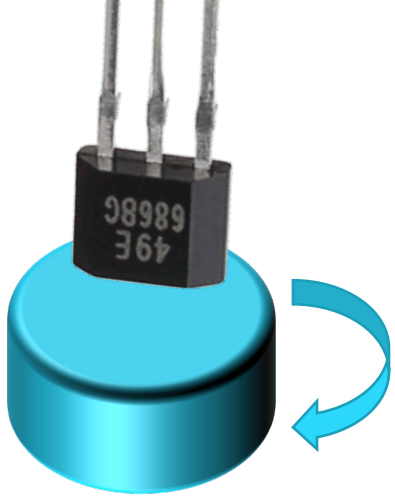
Mechanical Encoder
CPR: ~ 5 to 20
Cost: ~$1
Vendor: Bournes

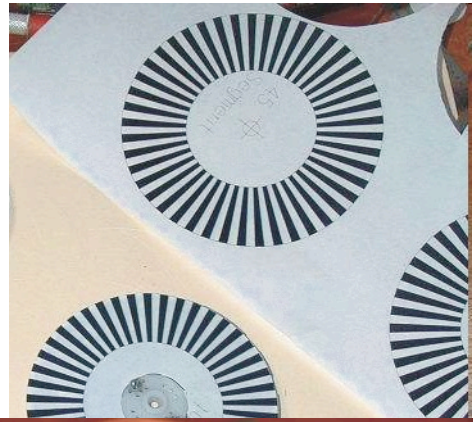
Optical Shaft Encoder
CPR: ~500 to 4000
Cost: ~$30-100
Vendor: Avago/Broadcom

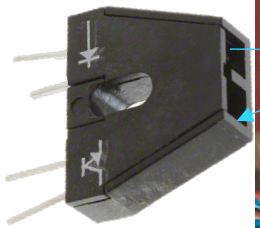
Magnetic Encoder
CPR: ~300-2000
Cost: ~$8
Vendor: AMS

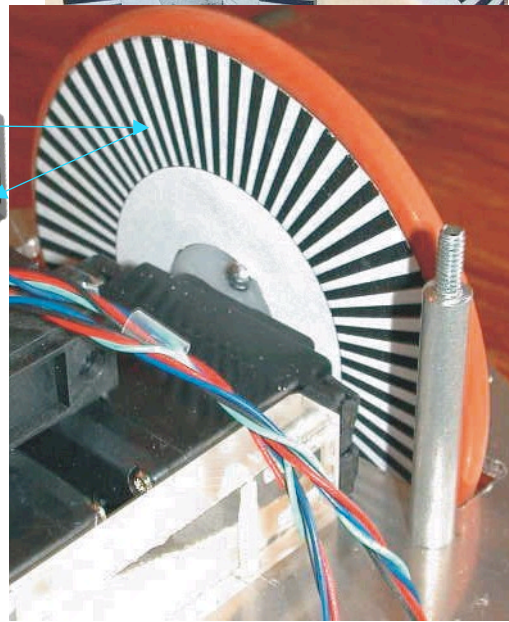
DIY Rotary Measurement

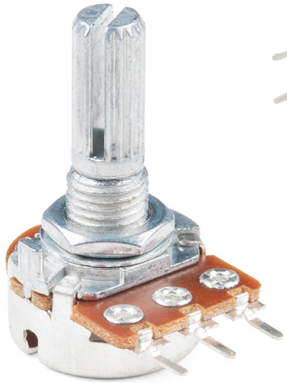
- Hall effect sensor (use ADC)
  - Absolute with ambiguity
  - Can't tell direction
  - Good for velocity sensing
- Make optical encoder disk
  - Need to be careful about ambient light
- Hack potentiometer
  - Break limit stops
  - Deal with gap in wiper











Rotary Shaft Encoders

Pros

- Robust to noise
- Quick to read
- Easy to make low res version
- Non contact (high life span)
- Very commonly used in industry

Cons

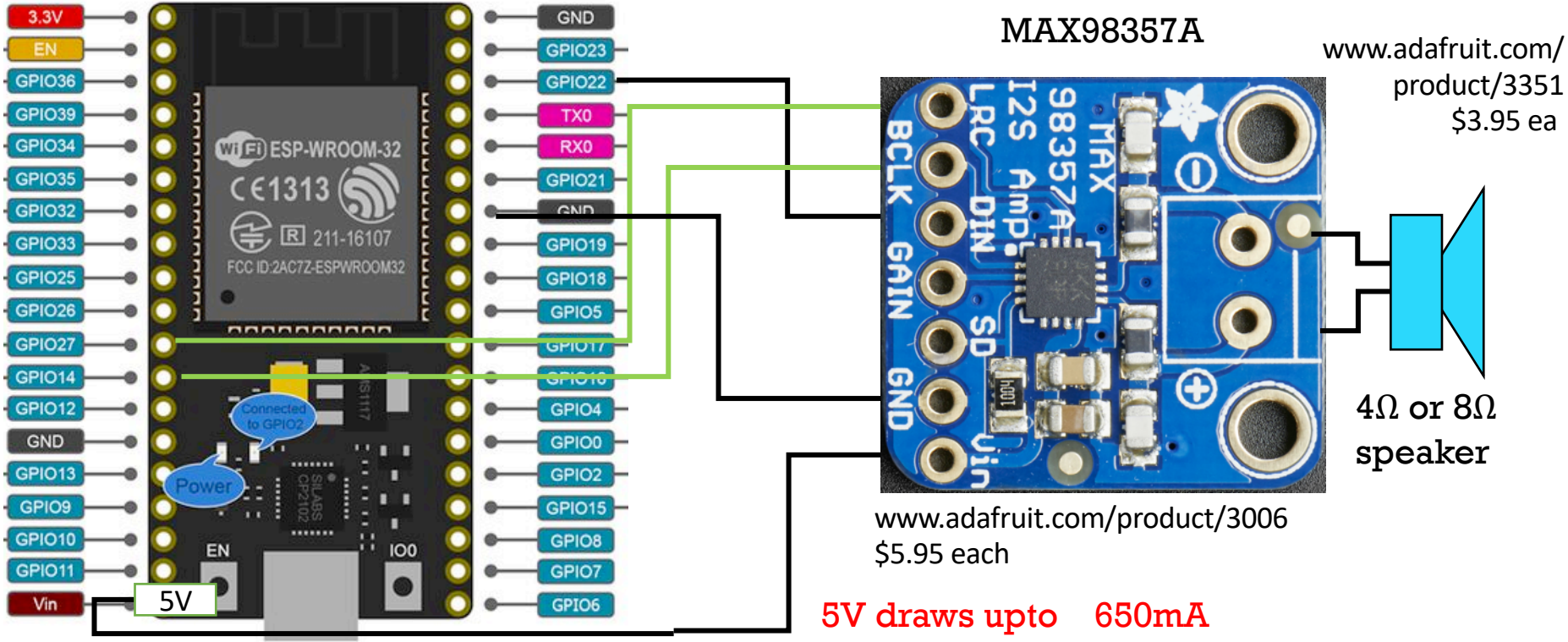
- Limited resolution
- Requires monitoring or losing position
- Expensive to make high res
- Sensitive to dust, impact.

03

Arduino Peripheral Example: Audio with I2S
Debugging Arduino

I2S on Arduino on ESP32

Arduino file on canvas -> files -> resources -> ESP32-i2sSounds



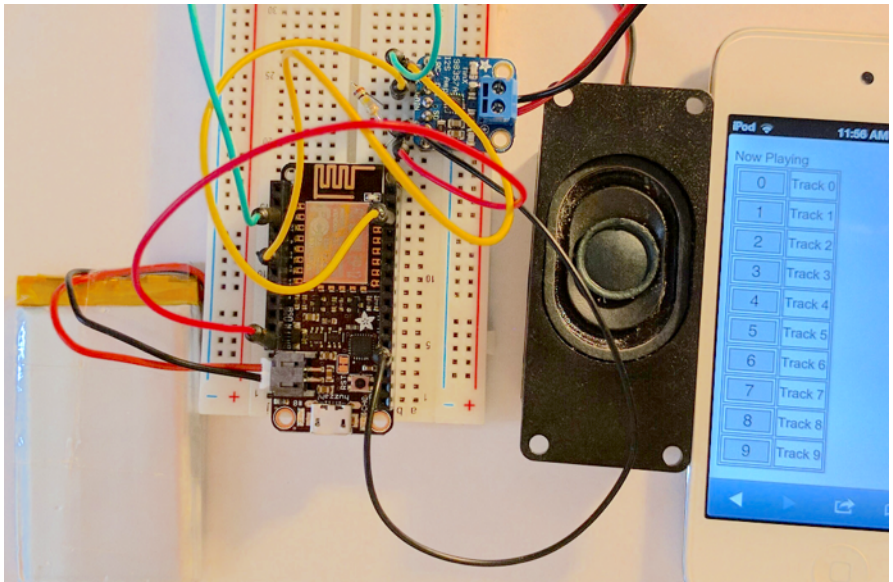
Navigating the Arduino Free Software World

1. You find some site that has software – download – test
2. Try to modify so it works for your case.
3. Find library that code relies on.

   http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use
4. Search for Library source code. (typically github)
5. Debug the mistakes from public domain software.
6. Learn and use new system.

Arduino I2S Example Scenario (from 2018)

- Google I2S ESP8266
  - https://github.com/bbx10/SFX-I2S-web-trigger
  - I2S from SPIFFS file plus web interface
- Set up and run
  - Works!
- Change to fit
  - Remove web stuff
  - Upload different sound files
- Result
  - Crashes sometimes
  - Doesn't play some sound files



Arduino I2S Example Scenario: Bug #1

- Problem: crashes
  - Play same 3 files over and over (in loop())
  - After 10 plays it crashes. (every time).
- Google I2S library
  - github.com/esp8266/Arduino/blob/master/cores/esp8266/core_esp8266_i2s.c
- Examination of I2S C code shows `i2s.begin()` uses `malloc()` and `i2s.end()` uses `free()`.
- The Arduino code calls begin() and end() for every song file.
  - malloc() and free() tend to fragment memory and overtime things crash.
  - Solution: call begin() once in setup() instead.

For embedded code – avoid malloc() and free(). Usually, you know how much space you need. Pre-allocate in a global.

Arduino I2S Example Scenario: Bug #2

- Problem: sometimes sound crackles or drops
  - Especially if moving wires.
- The I2S runs at high frequencies.
- The sound amp takes high currents (~700mA)
- Push on individual pins to see which is the problem.

Q10: What could be the issue?

- Solution: use better wire connections. Resolder pins.

Sometimes hardware is the problem – but usually suspect software first.

In general, software is consistent (except things like interrupts). If you have an intermittent problem, suspect a hardware issue.

Arduino I2S Example Scenario: Bug #3

- Problem: crashes
  - Simplifying cleaning up code…
  - Commenting out Serial.println() causes a crash.
- Must be a timing issue.
- Program writes bytes (e.g. a sound file) to a DMA (direct memory access) subsystem in big chunks. DMA writes those bytes one at a time to I2S port without processor babysitting.
- Problem DMA buffer was not clearing in time. Didn't think about wav_loop() calls happening quickly in succession.
- Solution: use library i2s_is_full() routine instead of own flags.

If removing println() makes something crash – suspect a timing issue.

Summary

- Sample Arduino code is everywhere. Quality is random.
- If you see random crashes for repeated code (without interrupts) suspect a memory leak or fragmentation. Look for overusing malloc() and free().
- If removing println() causes crashes, suspect a timing issue.
- If you have flakey hardware, push on wires to find suspect connections.

Answer in CHAT

Answer how you feel about each topic below with:

1. I don't understand this topic at all
2. I don't know now, but know what to do to get by
3. I understand some, but expect to get the rest later
4. I understand completely already

A. Hardware/software Debugging processes
B. Noise Sources and Reduction techniques
C. Encoders