Lab 6: System Identification

EECS 16B Spring 2024

Slides: http://links.eecs16b.org/lab6-slides

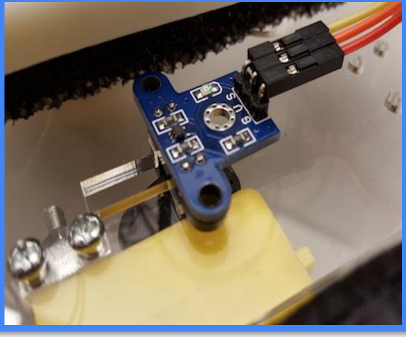
Administrivia

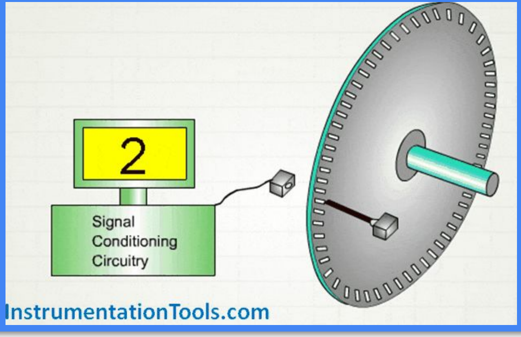
- If you are outside Cory 125 running your car, put both your computer number and “outside” for computer number field on help request
- Lab Grades error: https://links.eecs16b.org/lab-checkoff-error

Lab 6 Overview

- Sensor Verification
- Model Characterization
    - Data Collection: coarse and fine data
    - Linear Parameter Estimation: least-squares regression analysis
- Determine optimal operating velocity for the car
- Next lab
    - Controls: Design and test open-loop and closed-loop controllers

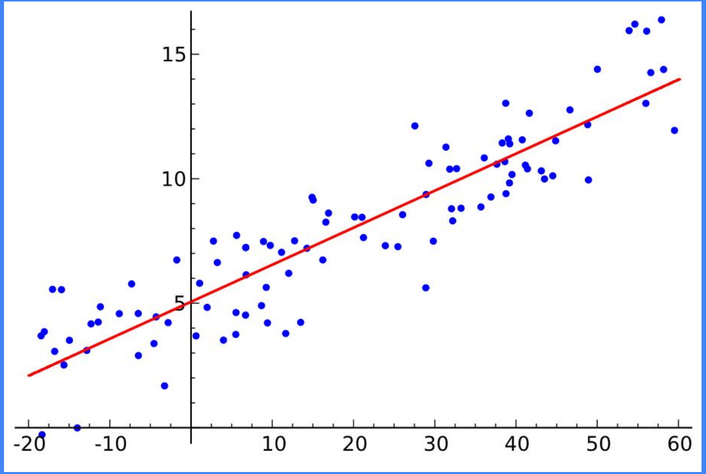
Review: Encoders





- Beam of light between 2 "legs"
  - outputs voltage based on whether the beam of light is blocked or unblocked
  - Mounted on "encoder wheels," which have many holes
    - As wheel rotates, spokes block and holes unblock the beam of light
- Can calculate velocity of car from rate of encoder value change
- 3 pins
  - **"G"** = ground
  - **"V"** = voltage (connect to breadboard positive rail, NOT Arduino's 5V pin)
  - **"S"** = encoder signal (connected to Arduino)

SIXT33N Car Model and Least-Squares



Source: thoughtco

Car Model

Left Side: $v_L[i] = d_L[i+1] - d_L[i] = \theta_L u_L[i] - \beta_L$
Right Side: $v_R[i] = d_R[i+1] - d_R[i] = \theta_R u_R[i] - \beta_R$

i - current timestep
v[i] - discrete time velocity
d[i] - total number of ticks advanced
u[i] - system input (in PWM, controlled by changing ***duty cycle***)
Θ - relates change in input PWM to change in velocity
β - velocity offset that encompasses real world imperfections like static friction

Read the lab note for how we solve for Θ and β and least-squares review!

$$\begin{array}{ccc} D_{data} & \vec{p} & \approx & \vec{s} \\ \begin{bmatrix} u[0] & -1 \\ u[1] & -1 \\ u[2] & -1 \\ \vdots & \vdots \\ u[\ell-1] & -1 \end{bmatrix} & \begin{bmatrix} \theta \\ \beta \end{bmatrix} & \approx & \begin{bmatrix} v[0] \\ v[1] \\ v[2] \\ \vdots \\ v[\ell-1] \end{bmatrix} \end{array}$$

Least Squares Review

- We rearrange our encoder model to resemble a linear equation:
  - Our equation takes the form: A**x** = **b**
  - We can solve this equation using Linear Least Squares, to find the best fit parameters

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$$

$$\vec{p} = (D_{data}^T D_{data})^{-1} D_{data}^T \vec{s}$$

  - We know u[i] and v[i], but want to find θ and β
- Numpy has helpful built-in functions:
  - Numpy.linalg.lstsq and numpy.vstack/numpy.hstack -> look at the documentation!
  - Transpose arrays with array_name.T

Determine Operating Point

$$v_L[i] = d_L[i+1] - d_L[i] = \theta_L u_L[i] - \beta_L$$
$$v_R[i] = d_R[i+1] - d_R[i] = \theta_R u_R[i] - \beta_R$$

- We measure v, we know u (that's our input PWM)
  - We can find θ and β from least squares
- Determine operating velocity point: What v* should we use? Make sure you check that the chosen v* works well with your model!
- Looking ahead to next lab … open-loop control
  - We can figure out the input u we need to set to achieve a target velocity v*
  - Does open-loop control work well for systems with disturbances?

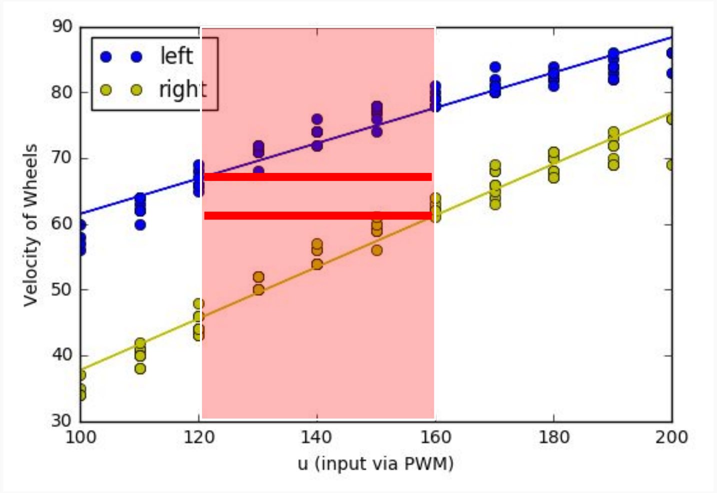
Letting Your Car Run Free

Collecting “Coarse” Data

- Let car run outside and collect data
  - After a brief delay, the car will begin driving
    - Arduino LEDs countdown
    - Power the circuit when you are outside
  - The Arduino will sweep through a wide range of PWM values
    - Should see it start fast, slow down, and speed back up before stopping
  - Car will not drive straight (most of the time)
- After finishing, upload data from Arduino to your computer
  - All 3 Arduino LEDs should blink to indicate that data is available for download
  - DO NOT unplug the Arduino Vin and plug in the USB (yes, at the same time)
  - Type in anything to serial monitor and hit enter to see your data printed

Collecting Fine Data

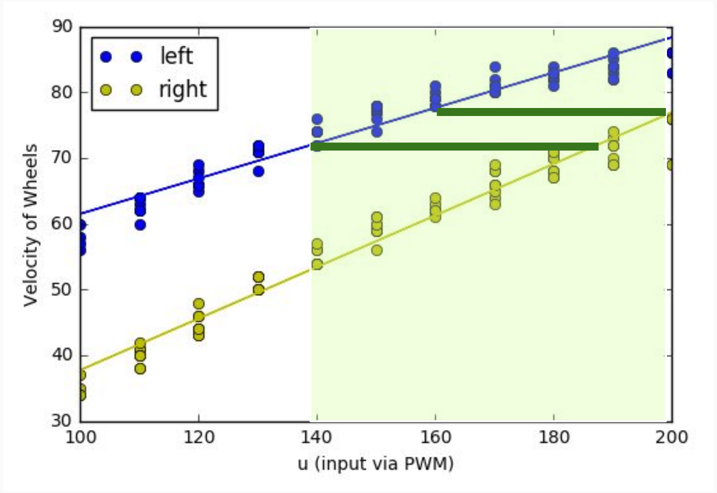
- After collecting “coarse” data, we will zoom into a linear range– where we can model the velocity response to PWM using linear parameters
- Choose a range of PWM values where both wheels can reach the same velocity for some PWM within the range



No overlapping range!

Collecting Fine Data

- After collecting “coarse” data, we will zoom into a linear range– where we can model the velocity response to PWM using linear parameters
- Choose a range of PWM values where both wheels can reach the same velocity for some PWM within the range



Overlapping range! Yay!

Powering the Car and Arduino

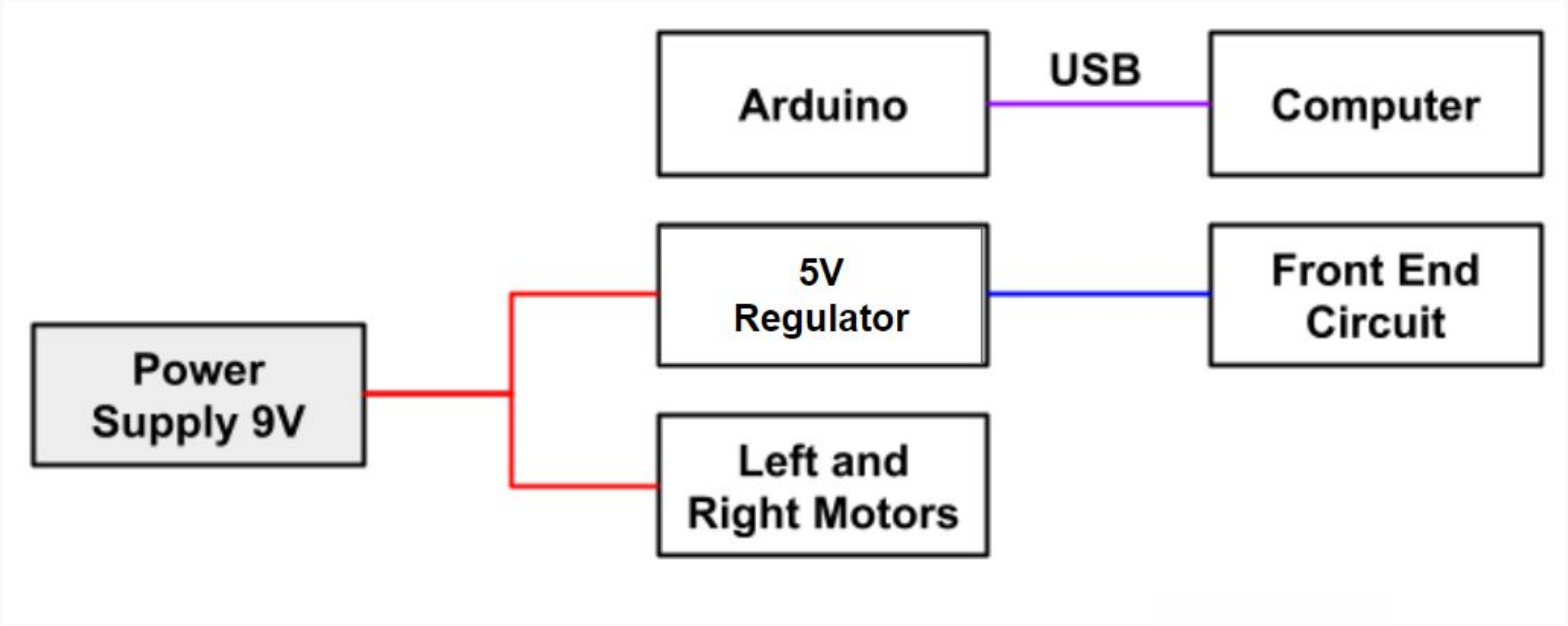
Powering your car

- Use two 9V batteries
  - One for regulator circuit (and sometimes Arduino), one for motors
  - When using the batteries, the RED is 9V and BLACK is GND
- When at the lab benches, use power supply and NOT batteries
  - Save batteries for when you're letting the car drive around
  - You'll need to replace batteries if they drain too low (<~7V or when motors stop running)
- Unplug the batteries when not in use to avoid draining them!
- Remove battery clips after lab to avoid accidental shorts!

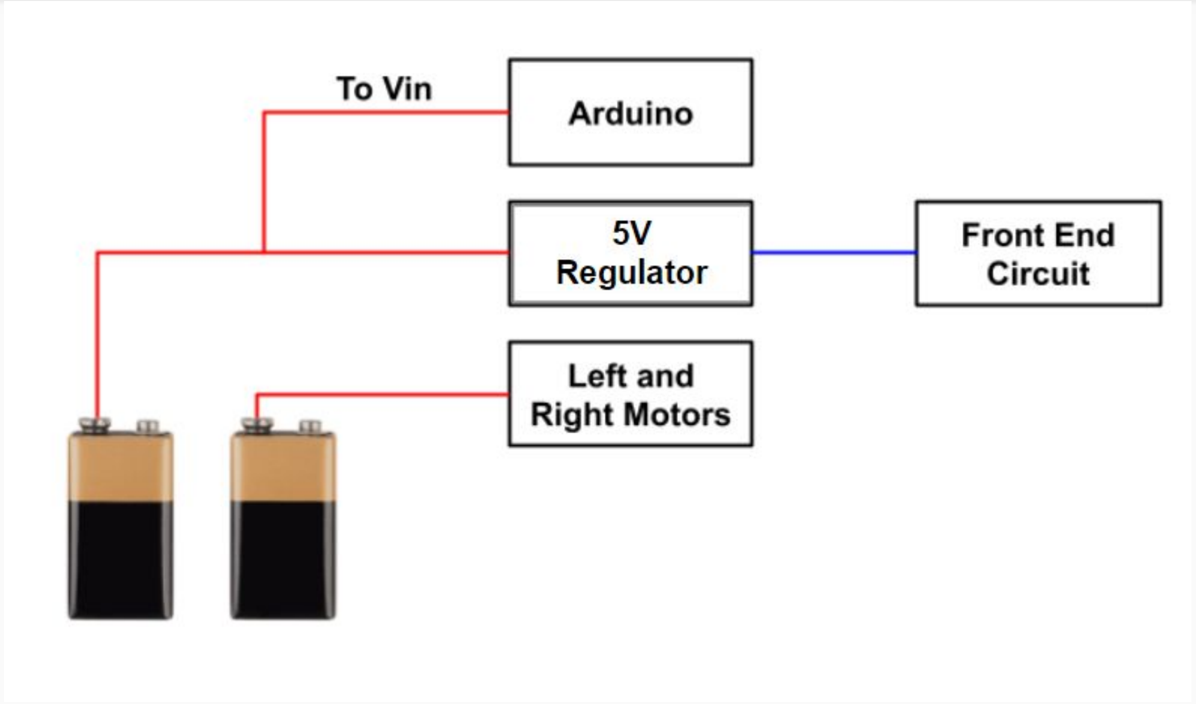
Powering your Arduino

- Arduino has 2 input power options: USB and 7-12V pin
  - Tethered: Use USB when you’re uploading code and downloading data
  - Mobile: Use Vin pin (connected to the same 9V as the regulator) when car is driving around
- **Ensure that you plug in BOTH USB and Power when collecting data**

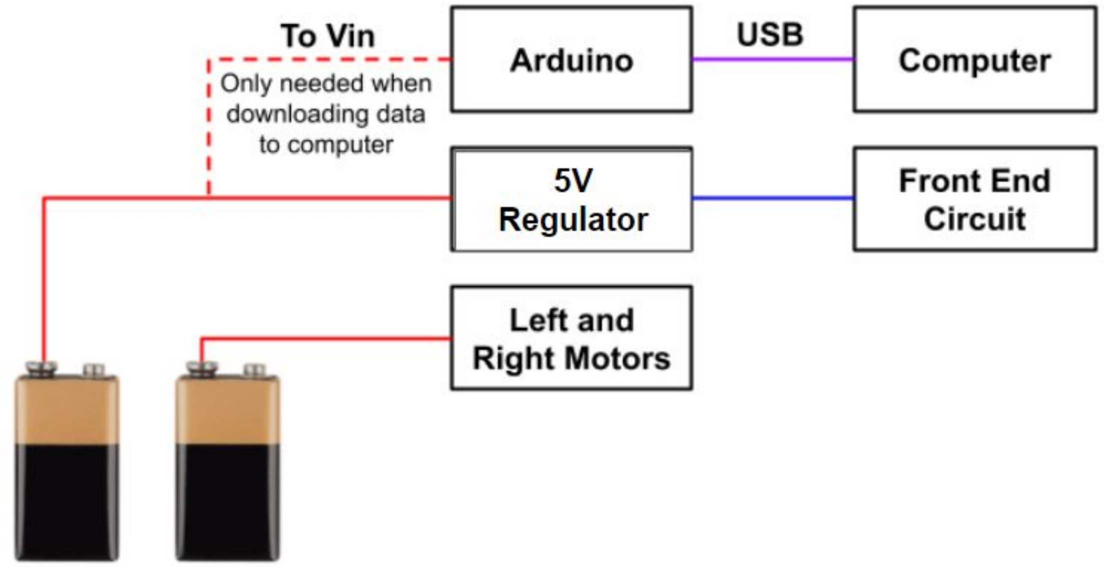
Tethered Powering: uploading code



Mobile Powering: driving car around



Data Recovery: downloading data

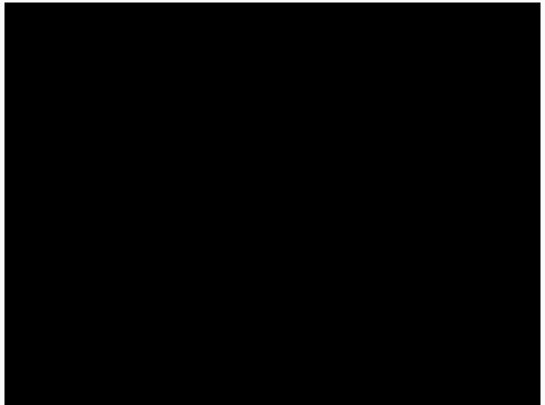


Analyzing Data

- How well does your model fit your data?
  - Do the lines look like they match up with the dots?
  - Do the velocities of the wheels make sense?
  - Are there different ranges of velocities where our linear model fits better than others?
- Common Bugs
  - Data is flat despite wheels turning → rerun encoder tests
  - Isolate issues by using symmetry to your advantage if one side works → swap components to see if it is a circuit, wire, encoder, or Arduino pin issue

Tips, Tricks, and Warnings

- Collect data in wide, flat area (hallways outside Cory 125)
  - Try to reposition car so that it doesn't hit any walls
  - If car is going to hit a wall, quickly pick it up and change its direction before it collides
- Car's orientation
  - **When the car is moving, the castor wheel should always be at the back of the car**



Important Forms/Links

- Help request form: https://eecs16b.org/lab-help
- Checkoff request form: https://eecs16b.org/lab-checkoff
- Anon Feedback: https://eecs16b.org/lab-anon-feedback
- Lab Grades error: https://links.eecs16b.org/lab-checkoff-error