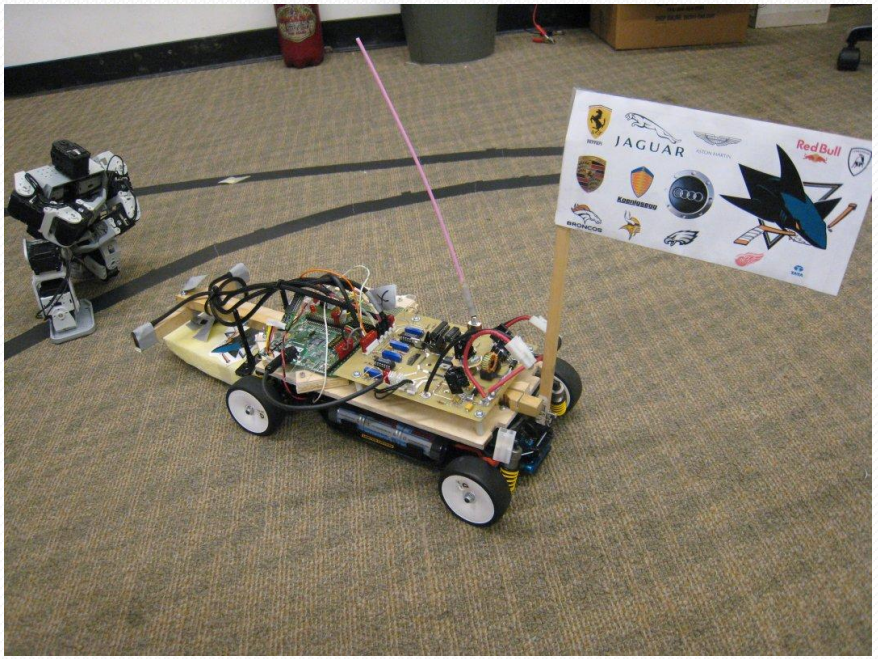
Team #2 - FFHDAM

EE192 Spring 2009 – Final Presentation

Farzad Fatollahi-Fard
Hartej Dhami
Aman Mouhidin

Overview

- Hardware
  - General
  - Sensors
- Software
- Controls
- Performance
- Roles and Contributions



Hardware - General

- We used ADUC7026 evaluation board
  - Pros

    We had cross-talk in adjacent ADC pins, with 11 ADCs we were able to use every other

    Easy to change GPIO functions (All pins have headers)
  - Cons

    Accommodate larger board (5-6 times more area than mini-board)

    Always get asked "Why is your board so big?"
    - We were assigned the ADUC7026, we just stuck with it.
- We used hi-torque instead of hi-speed servo
  - Pros

    Turns not limited by lateral momentum
  - Cons

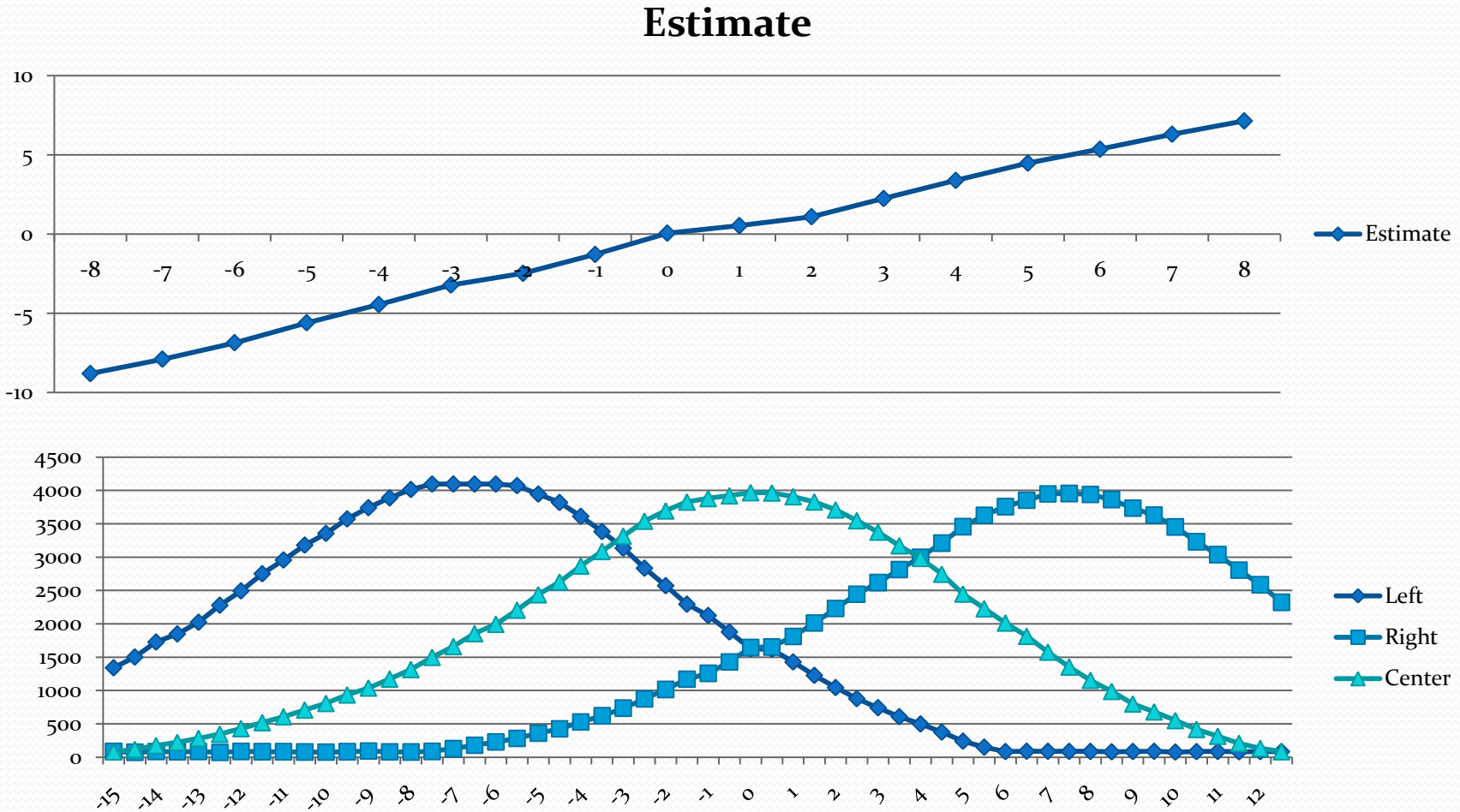
    Lower slew rate (0.08 $^{rad}/_{s}$ vs. 0.06 $^{rad}/_{s}$)

Sensors

- We used three sensors
  - Middle, Left, and right
  - PCB can handle four sensor inputs if need be
- We used a two stage amplifier
  - One gain stage and One buffer stage
- Our power circuitry and sensor circuitry had common ground plane
  - We did not run into any noise problems with a common ground plane
  - The DC boost and sensor op-amps were on opposite parts of the PCB

Sensors

Charts



Software

- Optimizations
  - We tried to do bit shifting if possible
  - Common conditions are at top of if statement tree
  - Used shorts instead of integers
- There is no 'special' feature
  - We believed that simpler code would result in best performance (KISS, more knobs…)
- Able to change k_p & k_d via Bluetooth™

Software

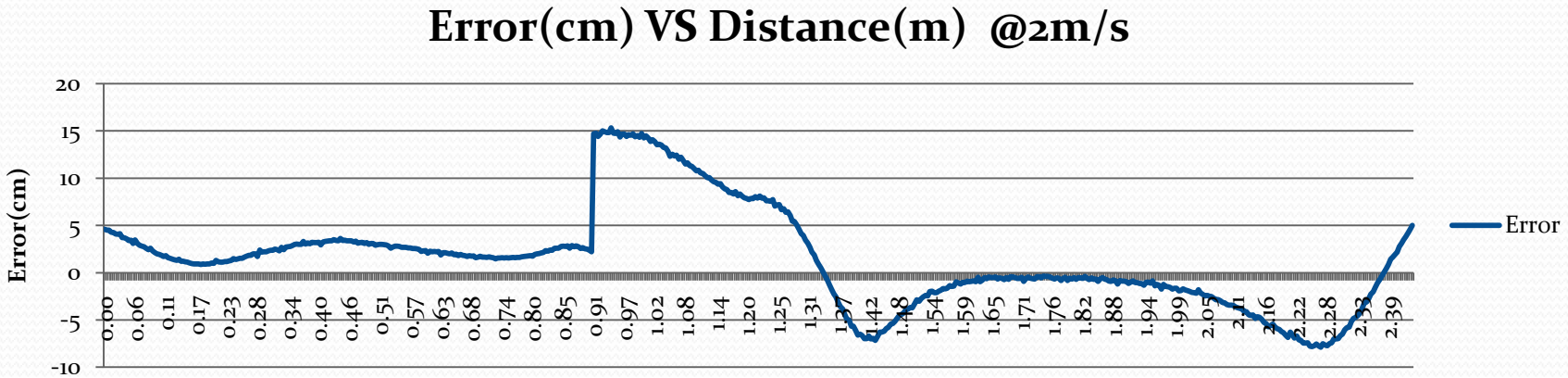
- Locked in last turn direction to recover if track is lost
- Used saturation on servo turn angle calculation to protect against large current draw from V-REG
- Used timers for servo PWM and in-built PWM for motor
- Used version control for code (SVN)
  - Better is not the enemy of good with version control for code anymore

Control

- We used a PD controller
  - centerServo + (k_p × diff)/100 + (k_d × (diffL - diff))/100;
- We started with an estimate for the k_p and k_d values
- Then we changed k_p and k_d via Bluetooth™ until we saw a stable result
- Initial Problems:
  - Initially we were sampling the ADC values in the while loop, every x^{th} iteration of the loop
  - Our car was very wobbly and this was because we were not sampling often enough
  - We changed our code to sample ADC values based on Timer2 interrupt
- Simulations proved helpful in understanding the concepts.
  - We tried doing simulation to get a starting value for k_p and k_d, those values were close but a little off from what we ended up using
  - k_p: 5.54 $^{rad}/_{m}$; k_d: 1.82 $^{rad \times s}/_{m}$

Performance

- Runs well except following short comings:
  - Twitches on straight-ways at crossings
  - Slips on turns
  - Tries to run away at 100% Duty Cycle PWM when touched inappropriately



Things we tried

- Use cross sensor
  - If cross value high enough ignore other sensors and keep last servo value
  - Decreased performance

    Car twitches a lot more.
- Speed encoder
  - Simple addition/subtraction scheme to modulate speed
    - Held speed within 0.5 ft/s of requested speed
  - Worked fine indoors
  - Did not work in sunlight

Roles and Contributions

- Hartej
  - Software Development
  - Car platform/boom design
  - PCB Schematic & Layout 2
- Farzad
  - Software Development
  - PCB Schematic & Layout 1 and 2
- Aman
  - Circuit Design
  - PCB Schematic & Layout 1 and 2

Questions?

Fin