EECS192 Lecture 4
Velocity and Line Sensor I
Feb. 11, 2020

Topics
• Driving MOSFETs and motor (conclusion)
• Battery Model
• Power wiring
• Driving MOSFETs and motor (conclusion)
• Quiz 1
• Latchup gotcha
• Speed sensing
• Line sensor

Notes:
Check off-
• 2/14: Motor drive/stall, steering servo using LiPo
• Quiz 2: power MOSFET/motor drive Tues 2/18
Topics

- Driving MOSFETs and motor (conclusion)
- Battery Model
- Power wiring
- Quiz 1
- Latchup gotcha
- Speed sensing
- Line sensor
Motor Electrical Model

Back EMF
Motor electromechanical behavior

Also- see motor worksheet……

\[ i_m = \frac{V_{BAT} - k_e \dot{\theta}_m}{R_m} \]

Conclusion:
\[ <i_m> = ? \]

Motor Resistance?
Peak current?
Driving MOSFETs and motor

\[ R_m = 0.1 \text{ ohms}, \quad V_{\text{batt}} = 7.2 \text{ V}, \quad R_{\text{bat}} = 0. \]
\[ V_{\text{ds}} = 3.6 \text{ V} \quad \Rightarrow \quad I_{ds} = \frac{(7.2-3.6) \text{ V}}{0.1 \text{ ohm}} = 36 \text{ amps} \]

\[ V_{\text{gs}} = 20 \text{ V} \quad \Rightarrow \quad P_{\text{trans}} \sim 35 \text{ W} \]
\[ V_{\text{gs}} = 6 \text{ V} \quad \Rightarrow \quad P_{\text{trans}} \sim 72 \text{ W} \]

Key design points:
1) High \( V_{\text{gs}} \) better than low \( V_{\text{gs}} \)
2) Switch quickly
3) Make sure \( V_s = 0 \) (big ground)
Topics

• Driving MOSFETs and motor (conclusion)
• Battery Model
• Power wiring
• Quiz 1
• Latchup gotcha
• Speed sensing
• Line sensor
Battery Model- 3S
avoid weakly charged cell

\[ i \]
\[ R_{\text{load}} \]

3.7V

-3.7V
Topics
• Driving MOSFETs and motor (conclusion)
• Battery Model
• Power wiring
• Quiz 1
• Latchup gotcha
• Speed sensing
• Line sensor
On board: what does this look like electrically (as a schematic)?
On board: what does this look like electrically (as a schematic)?
1.4 Voltage and current operating ratings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Min.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{DD} )</td>
<td>Digital supply voltage</td>
<td>(-0.3)</td>
<td>3.3</td>
<td>V</td>
</tr>
<tr>
<td>( I_{DD} )</td>
<td>Digital supply current</td>
<td>(-0.3)</td>
<td>1.0</td>
<td>mA</td>
</tr>
<tr>
<td>( V_{IO} )</td>
<td>IO pin input voltage</td>
<td>(-0.3)</td>
<td>( V_{DD} + 0.3 )</td>
<td>V</td>
</tr>
</tbody>
</table>

Caution: input voltage from sensor may be greater than 0.3V when CPU is off \( V_{DD} = 0! \)
Latchup phenomena: make sure $V_{in}$ always less than $V_{dd}$

![Protection circuit diagram](image1)

![Schottky diode](image2)

![Protection circuit diagram](image3)
Topics

- Driving MOSFETs and motor (conclusion)
- Battery Model
- Power wiring
- Quiz 1
- Latchup gotcha
- Speed sensing
- Line sensor
Quadrature Encoder


3.3 V DC

0 V DC

A

B

Fab suggestion: aluminum foil covered with black paper with slots. 4 slots probably enough. Note: sensors can be placed where convenient—don’t need to look at same slot.

http://codeforfree.weebly.com/uploads/1/0/1/6/10160088/5858962_orig.png
Beagle Bone Blue Quad Encoder

```c
int rc_encoder_read  (int ch)
```

Returns
The current position (signed 32-bit integer) or -1 and prints an error message if there is a problem.

Ch 1-3 are available

Examples:
```c
rc_test_encoders.c.
```
Sharp GPS260

- Choose current 4 mA in LED
- Vcc = 3.3 V
- May want regulated/clean voltage for Vcc

100 us response time
Velocity Sensing

• On board: estimating $\Delta x/\Delta T$

Note: care about velocity sensing usually at cruise speed (also stopping)
Topics
• Driving MOSFETs and motor (conclusion)
• Battery Model
• Power wiring
• Quiz 1
• Latchup gotcha
• Speed sensing
• Line sensor
TSL 1401 line sensor

Functional Block Diagram
TSL 1401 line sensor

PARAMETER MEASUREMENT INFORMATION

Figure 1. Timing Waveforms

Figure 2. Operational Waveforms
Possible algorithms for line detection

e.g. scipy.signal.filter

• Subtraction- to find left and right edge of line (ok if not noisy, somewhat lighting invariant)

• Difference of gaussians (idea is to smooth then differentiate)

• Correlation (best match position for known features)
  – scipy.signal.correlate
TSL 1401 line sensor NATCAR 8 bit

Frame 0

Frame 1

Frame 1-Frame 0

Frame 0

Frame 2

Frame 2-Frame 0

inner step cir1800-2300.csv
Programmable Real-time Unit (PRU)

Shared RAM used for communication with Linux process

Used on BeagleBone Blue
PRU1: RC servo
PRU0: real-time A/D for line camera (was originally used for quad encoder Ch4)
## Shared 12 kb Address Space

<table>
<thead>
<tr>
<th>PRU0</th>
<th>LINUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1_0000</td>
<td>For encoder</td>
</tr>
<tr>
<td>Flag = 1 to start conversion</td>
<td>rc_pru_shared_mem_ptr();</td>
</tr>
<tr>
<td>Pixel 0</td>
<td>shared_mem_32bit_ptr[16+1]</td>
</tr>
<tr>
<td>...</td>
<td>shared_mem_32bit_ptr[16+2]</td>
</tr>
<tr>
<td>Pixel 127</td>
<td>shared_mem_32bit_ptr[16+2+127]</td>
</tr>
</tbody>
</table>

**Linux LineCamera.c line 116:**

```c
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 1;
// set flag to start conversion by PRU
while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 1);
```

**PRU main_pru0.c line 117:**

```c
while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 0);
// loop until command
//... read 128 pixels ...
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 0;
// reset to zero
```
Extra Slides
Velocity control overview

On board...
Proportional control:
\[ U = kp^*e = kp^* (r - y); \]
Here: \( r \) is desired velocity, \( U \) is PWM %

Proportional + integral control
\[ U = kp^*e + ki^* e_{\text{sum}}; \]
\[ e_{\text{sum}} = e_{\text{sum}} + e; \]
Velocity sensing

\[ \theta(t) \]

\[ V_{\text{uniform}} = \frac{N \Delta \theta}{10 \text{ ms}} \]

\[ V_{\text{edge}} = \frac{\Delta \theta}{\Delta T_1} \]

Edge timing

Uniform sampling

\[ \Delta T_1 \]

\[ \Delta \theta \]

\[ 10 \text{ ms} \]
Velocity sensing (recap)

\[ V \sim \frac{\text{change in angle}}{\text{change in time}} \]

On board…
Non-inverting amp to measure motor current. Back EMF can be estimated from battery voltage and motor resistance.

1.8V

To A/D

Low voltage r-r op-amp

<table>
<thead>
<tr>
<th>Cu Weight oz.</th>
<th>Thickness mm(mils)</th>
<th>mΩ/Square 25°C</th>
<th>mΩ/Square 100°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>.02 (0.7)</td>
<td>1.0</td>
<td>1.3</td>
</tr>
<tr>
<td>1</td>
<td>.04 (1.4)</td>
<td>0.5</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Back EMF velocity sensing
Analog/Digital Overview

Figure 12-2. Functional Block Diagram

Caution: 1.8V MAXIMUM
Note: lots of sequencing/delays.
Set up in PRU0 to read 128 elements at ~8 us/sample.

(1) In the device-specific datasheet:
- VDDA_ADC and VSSA_ADC are referred to as "Internal References"
- VREFP and VREFN are referred to as "External References"
TSL 1401 line sensor - option exposure control
(would need to modify PRU code)

Exposure time (>25 us)

SI

129 clocks

CLK

A/D read ~12 us

AO

Clock only
no A/D read
Automatic Gain Control

In all the discussion that follows, we will be using one-shot imaging.