# EECS 192: Mechatronics Design Lab

## Discussion 11: Tips

GSI: Justin Yim

10 & 11 April 2019 (Week 11)

- Tips
- Automatic Gain Control

# Integration Troubles

- Car integration problems
  - BBBL dies on power loss

- Potential solutions

# Integration Troubles

- Car integration problems
  - BBBL dies on power loss

- Potential solutions
  - 5V power *may* (link) be safer

# Integration Troubles

- Car integration problems
  - BBBL dies on power loss
  - How to share grounds when the BBBL is powered over USB on the bench?

- Potential solutions
  - 5V power *may* (link) be safer

# Integration Troubles

- Car integration problems
  - BBBL dies on power loss
  - How to share grounds when the BBBL is powered over USB on the bench?

- Potential solutions
  - 5V power *may* (link) be safer
  - Make a benchtop harness that connects to the battery port (so it can't be left in)

# Integration Troubles

- Car integration problems
  - BBBL dies on power loss
  - How to share grounds when the BBBL is powered over USB on the bench?
  - Board fries when it is first powered on

- Potential solutions
  - 5V power *may* (link) be safer
  - Make a benchtop harness that connects to the battery port (so it can't be left in)

# Integration Troubles

- Car integration problems
  - BBBL dies on power loss
  - How to share grounds when the BBBL is powered over USB on the bench?
  - Board fries when it is first powered on

- Potential solutions
  - 5V power *may* (link) be safer
  - Make a benchtop harness that connects to the battery port (so it can't be left in)
  - Methodical board bring-up: verify system modules are working in isolation (verify expected signals before applying full battery power, etc.)

# Motor Troubles

- Problem: circuits behave differently with motor attached. Why?

Image from (link)

# Motor Troubles

- Problem: circuits behave differently with motor attached. Why?
- The motor draws a lot of current and generates a lot of EMI. How to debug?

Image from (link)

# Motor Troubles

- Problem: circuits behave differently with motor attached. Why?
- The motor draws a lot of current and generates a lot of EMI. How to debug?
- Check line resistance with multimeter and check noise with oscilloscope. What are some design fixes?

Image from (link)

# Motor Troubles

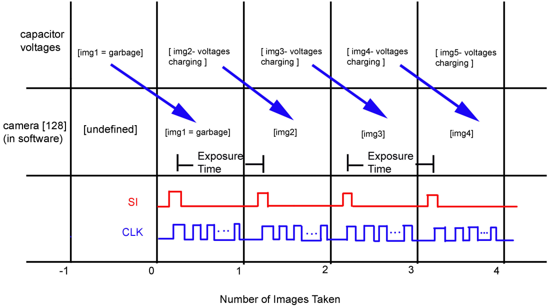
- Problem: circuits behave differently with motor attached. Why?
- The motor draws a lot of current and generates a lot of EMI. How to debug?
- Check line resistance with multimeter and check noise with oscilloscope. What are some design fixes?
- Thick traces & wires for low resistance, better shielding (ground planes, filter caps, diodes, cable assembly)

Image from (link)

# Automatic Gain Control

- So the lighting on the 3rd floor is different than in the lab?
- Solutions
    - External Lights (LED, flashlights, etc.)
    - Robust line detection (derivatives, LPF, cross correlation- see discussion 8)
    - Automatic Gain Control!!

# TSL1401 Timing- No Automatic Gain Control

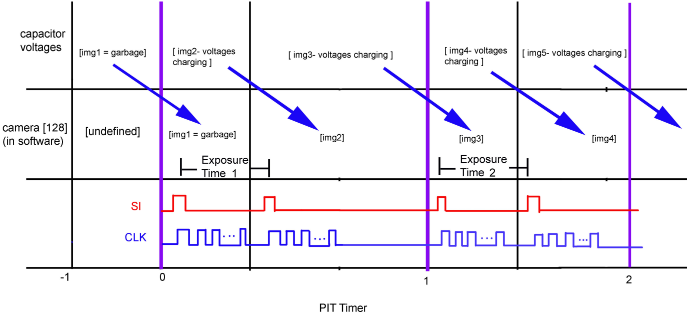


# Pseudocode (PRU)

```
void take_pic(){
  SI High;
  CLK High;
  SI Low;
  for(i=0 to 128){
    CLK High;
    camera[i] = read_adc();
    CLK_Low;
  }
}
```

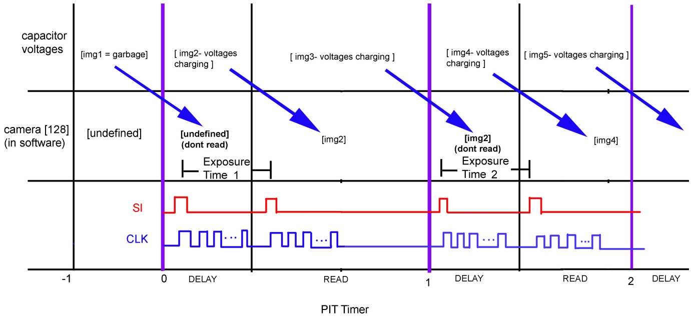
- Each call to take pic reads out the previous capacitor voltages
- There is currently no exposure control

# TSL1401 timing


capacitor voltages
[img1 = garbage]
[ img2- voltages charging ]
[ img3- voltages charging ]
[ img4- voltages charging ]
[ img5- voltages charging ]
camera [128] (in software)
[undefined]
[img1 = garbage]
[img2]
[img3]
[img4]
Exposure Time 1
Exposure Time 2
SI
CLK
-1
0
1
2
PIT Timer

- Don't need to read garbage frames!

# TSL1401 timing



- Don't need to read garbage frames!
- Removing read_adc speed's up code execution significantly

# Pseudocode (PRU)

```
void take_pic(int mode){
  SI High;
  CLK High;
  SI Low;
  for(i=0 to 128){
    CLK High;
    if (mode == 1)//Read
      camera[i] = read_adc();
    CLK_Low;
  }
  if (mode == 0)//Delay
    delay(camera_delay);
  else//Read
    adjust_camera_delay();
    // How might you do this?
}
```

```
void take_agc(){
    /* Clock out
    garbage data
    & expose new image */
    take_pic(0);
    /* Read new image
    and update exposure
    delay */
    take_pic(1);
}
```

# Code Structure v1 (Linux)

```
int main(){
    take_agc();
    find_line();
    estimate_velocity();
    calculate_new_controls();
    telemetry.do_io();
}

void interrupt_handler(){
   apply_servo_control();
   apply_motor_control();
}
```

- Pro- interrupt executes very quickly- potentially easier to debug
- Con- Potentially updating servo/motor control on old sensor readings

# Code Structure v2 (Linux)

```
int main(){
    take_agc();
    find_line();
    estimate_velocity();
    calculate_new_controls();
    apply_servo_control();
    apply_motor_control();
    telemetry.do_io();
}
```

- Pro- Updating servo/motor control on newest sensor readings
- Con- No interrupt to enforce timing