EECS192 Lecture 3
Feb. 2, 2021

Notes:

- 2/9 project proposal – upload to bcourses by Tues 5 pm
- Survey for Cory Courtyard

Topics

- Checkpoints 1,2,3
- Memory Usage build/runtime
- Updated Skeleton code tour
- Peripheral interface: A/D, pulse count (intro)
- Sensor: line camera

CP1- Measuring Timing from ESP32

High resolution timing using built-in 64 bit counter

```
uint64_t task_counter_value1, task_counter_value2;
timer_get_counter_value(TIMER_GROUP_0, TIMER_0,
      &task_counter_value1);
/* code to be timed here */
timer_get_counter_value(TIMER_GROUP_0, TIMER_0,
      &task_counter_value2);
```

Good/Badnews:

snprintf(log, sizeof(log), "Hello World \n"); ➔ ~ 20 us

snprintf(log, sizeof(log), "uint64_t %llu\n", task_counter_value); ➔ 81 us

but

snprintf(log, sizeof(log), "starttime %8.3f usec \n", 1e6*starttime); ➔ 633 us

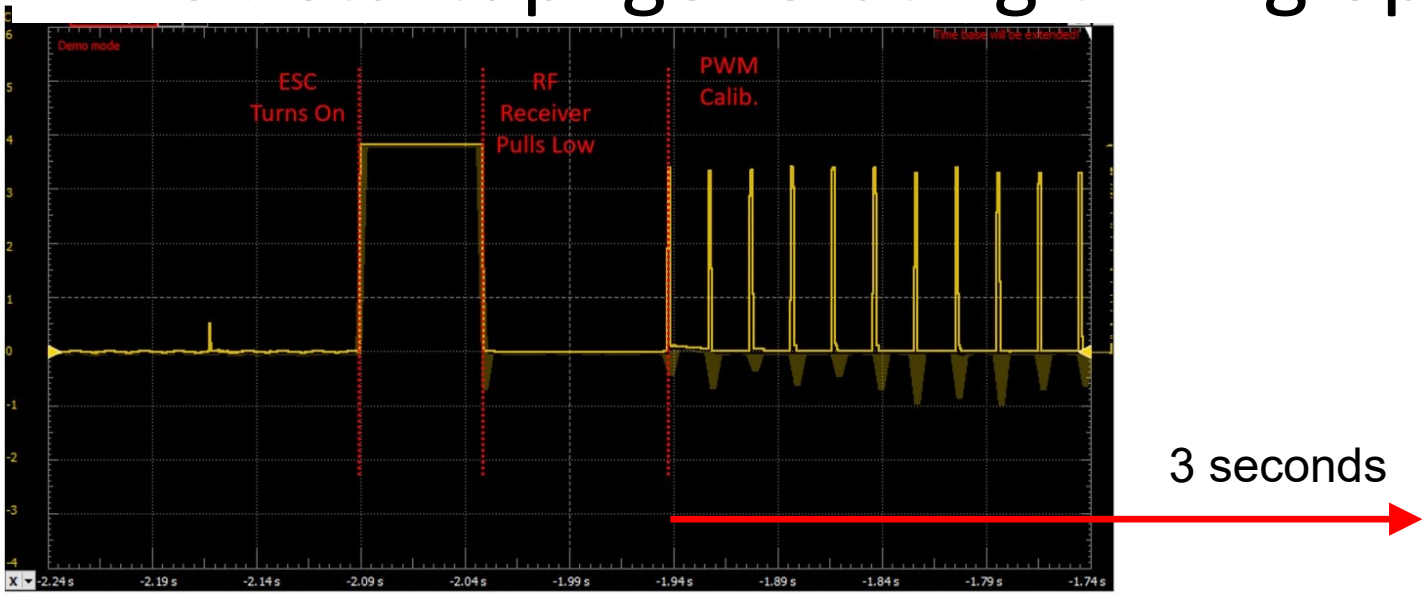
Maybe send double instead of text?

CP2- PWM for driving steering servo and ESC

Write code which performs the following sequence of functions:

- C2.1: Start wheels turning, and ramp up to full speed in 5 seconds and down to zero speed in another 5 seconds.
- C2.2: Set steering angle approximately half full-left and hold for 5 seconds. (For example, if full steering range is +- 20 degrees, set steering angle to +10 degrees.)
- C2.3: Set steering angle straight and hold for 5 seconds.
- C2.4: Set steering angle approximately half full-right, and hold for 5 seconds.
- C2.5: Set steering angle back to approximately straight.
- C2.6: Show steering changing and wheels turning at the same time
- C2.7: Report Data RAM and Instruction RAM usage. How much of each is left? [pio run –v in terminal window]
- C2.8: All members must fill out the checkpoint survey before the checkoff close. Completion is individually graded

ESC startup- generating timing options



1. vTaskDelay(pdMS_TO_TICKS(3000)); [1 ms resolution]
   or

2. Busy wait (may trigger watch dog for long delays)
   while(task_counter_value < end_value)
   { timer_get_counter_value(TIMER_GROUP_0, TIMER_0, &task_counter_value); }

   or

3. Interrupt alarm [usec resolution] (see timer-group-example.c in SkeletonHuzzah32)

/* if timer interval > CONFIG_ESP_TASK_WDT_TIMEOUT_S, will get watchdog timeout*/
timer_set_alarm();
docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/timer.html

CP3- remote control with UDP

(Intended to be easy since project proposal due Tues 2/9)

The car should be upside down, or lifted off the ground so it does not move

CAUTION: setting PWM to 1.0 ms or 2.0 ms can drive servo to the end of its range and cause it to burn out.

- C3.1 From the remote client keyboard, send a speed command for the drive motor. Pick a range such as 0-100, and show that you can specify a range of values. The motor should remain turning until the next command.
- C3.2: Find the minimum ESC PWM value which causes the wheels to turn.
- C3.3: With motors turning, send an "Emergency Stop" command to the car (the motor should be turned off or braked, if available).
- C3.4: From the remote client , send a command to set the steering angle.
- C3.5: Find the range of steering angles for your car. (These values should be used for range checking in your code for the rest of the semester.)
- C3.6: Show that motor speed and steering can be set independently from the remote client keyboard.
- C3.7: Report Data RAM and Instruction RAM usage. How much of each is left?

```
From VS terminal > pio run –v
```

- C3.8: All members must fill out the checkpoint survey before the checkoff close. Completion is individually graded.

EECS192 Lecture 3
Feb. 2, 2021

Notes:

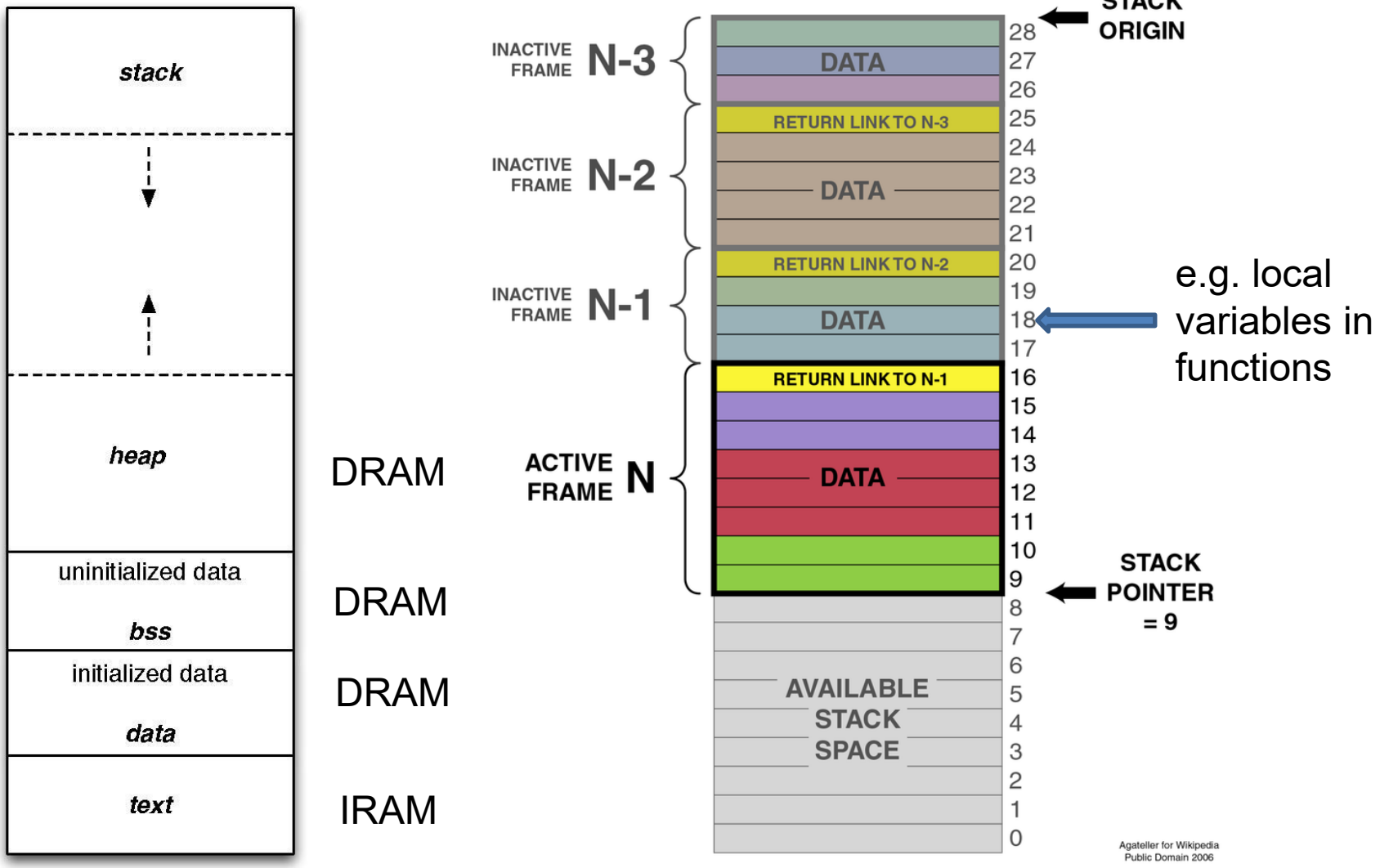
- 2/9 project proposal – upload to bcourses by Tues 5 pm
- Survey for Cory Courtyard

Topics

- Checkpoints 1,2,3
- Memory Usage build/runtime
- Updated Skeleton code tour
- Peripheral interface: A/D, pulse count (intro)
- Sensor: line camera

Recap: Intro to stack, heap, malloc, free, etc

Memory model



From Wikipedia:https://en.wikipedia.org/wiki/Data_segment

Memory use at compile time

From VS terminal > pio run –v

section	size	addr	
.iram0.vectors	1027	1074266112	IRAM
.iram0.text	88424	1074267140	
.iram0.text_end	0	1074355564	
.dram0.data	19912	1073414144	DRAM
.dram0.bss	24560	1073434056	
.flash.rodata	103292	1061158944	
.flash.text	486215	1074593824	
.debug_frame	91744	0	
.debug_info	3444763	0	
.debug_abbrev	287554	0	
.debug_loc	696445	0	
.debug_aranges	36008	0	
.debug_ranges	40656	0	
.debug_line	1623178	0	
.debug_str	283103	0	
Total	**7227598**		

Heap memory (in Data RAM)- Before tasks

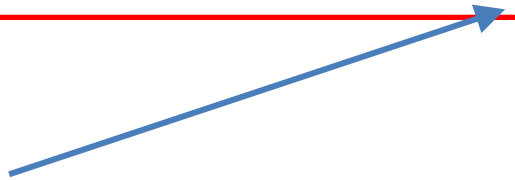
Used for stack, local variables, global variables
malloc() and free()

```
heap_caps_print_heap_info(MALLOC_CAP_8BIT);

Heap summary for capabilities 0x00000006:
  At 0x3ffae6e0 len 6432 free 0 allocated 6300 min_free 0
    largest_free_block 0 alloc_blocks 25 free_blocks 0 total_blocks 25
  At 0x3ffbada8 len 152152 free 136724 allocated 15284 min_free 135908
    largest_free_block 135908 alloc_blocks 26 free_blocks 2 total_blocks 28
  At 0x3ffe0440 len 15072 free 15036 allocated 0 min_free 15036
    largest_free_block 15036 alloc_blocks 0 free_blocks 1 total_blocks 1
  At 0x3ffe4350 len 113840 free 113804 allocated 0 min_free 113804
    largest_free_block 113804 alloc_blocks 0 free_blocks 1 total_blocks 1
  Totals:
    free 265564 allocated 21584 min_free 264748 largest_free_block 135908
```

Heap memory (in Data RAM)- after tasks

Heap summary for capabilities 0x00000006:
  At 0x3ffae6e0 len 6432 free 0 allocated 6300 min_free 0
    largest_free_block 0 alloc_blocks 25 free_blocks 0 total_blocks 25
  At 0x3ffbada8 len 152152 free 75752 allocated 75616 min_free 75368
    largest_free_block 75412 alloc_blocks 184 free_blocks 4 total_blocks 188
  At 0x3ffe0440 len 15072 free 15036 allocated 0 min_free 15036
    largest_free_block 15036 alloc_blocks 0 free_blocks 1 total_blocks 1
  At 0x3ffe4350 len 113840 free 113804 allocated 0 min_free 113804
    largest_free_block 113804 alloc_blocks 0 free_blocks 1 total_blocks 1
Totals:
  free 204592 allocated 81916 min_free 204208 largest_free_block 113804



Double is how many bytes? (8)
double track_data[20000] would overflow

70K allocated for tasks

EECS192 Lecture 3
Feb. 2, 2021

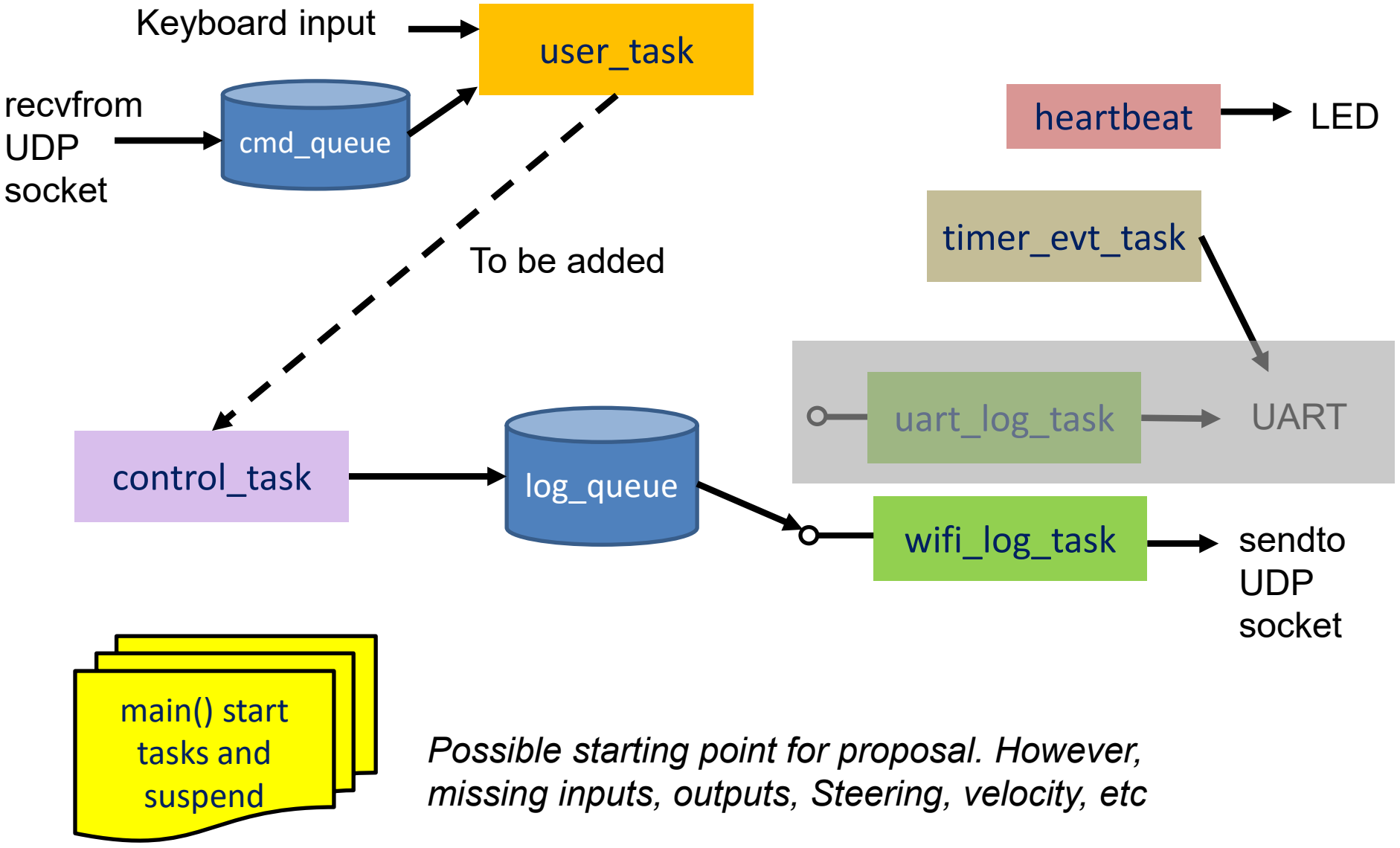
Notes:

- 2/9 project proposal – upload to bcourses by Tues 5 pm
- Survey for Cory Courtyard

Topics

- Checkpoints 1,2,3
- Memory Usage build/runtime
- Updated Skeleton code tour
- Peripheral interface: A/D, pulse count (intro)
- Sensor: line camera

SkeletonHuzzah32 Branch UDPCommd SW Block Diagram



Possible starting point for proposal. However, missing inputs, outputs, Steering, velocity, etc

Note conventions- data flow left to right

log-input-client-udp.py

```
msg = input('Enter string "command value": ')
temp= msg.split() # get separate command and value elements
 msg_bytes = struct.pack('8si',str.encode(temp[0]),int(temp[1]))
# use structure so there is no need for risky sscanf on Huzzah32
sock.sendto(msg_bytes, addr)  # send cmd + value over UDP
```

Example input: SPEED 120

Command is up to 8 characters, value is a 32 bit int

If python struct.pack is changed, need to also change skeleton.h

```
struct cmd_struct_def {
    char cmd[8];
    int value;};
```

wifi-log-server.c

```
size_read = recvfrom(sock,&cmd_struct,sizeof(cmd_struct),
0, (struct sockaddr *)&source_addr, &socklen);
if (size_read <= 0)
{ printf("socket read error\n");     }

printf("command: %s  value = %d \n",
      cmd_struct.cmd, cmd_struct.value);
xQueueOverwrite(cmd_queue, &cmd_struct);
// put command on queue, over writing any previous cmd
```

Example input: SPEED 120

cmd_struct.cmd = “SPEED”, cmd_struct.value = 120

```
struct cmd_struct_def {
    char cmd[8];
    int value;};
```

usertask.c

```
 // check for command waiting in queue
cmdready =  uxQueueMessagesWaiting(cmd_queue);
if (cmdready >= 1)  // get item from cmd queue
{ if(xQueueReceive(cmd_queue, &cmd_struct,
            portMAX_DELAY) != pdPASS)
  { printf("error reading from cmd_queue\n"); }
  udp_cmd(cmd_struct.cmd, cmd_struct.value);
 }
------------------------------
// handle commands from UDP
void udp_cmd(char command[], int value)
{  if (strcmp(command,"TIME") == 0)
        get_time();
  …
```

Add other commands here, use string compare to parse

EECS192 Lecture 3
Feb. 2, 2021

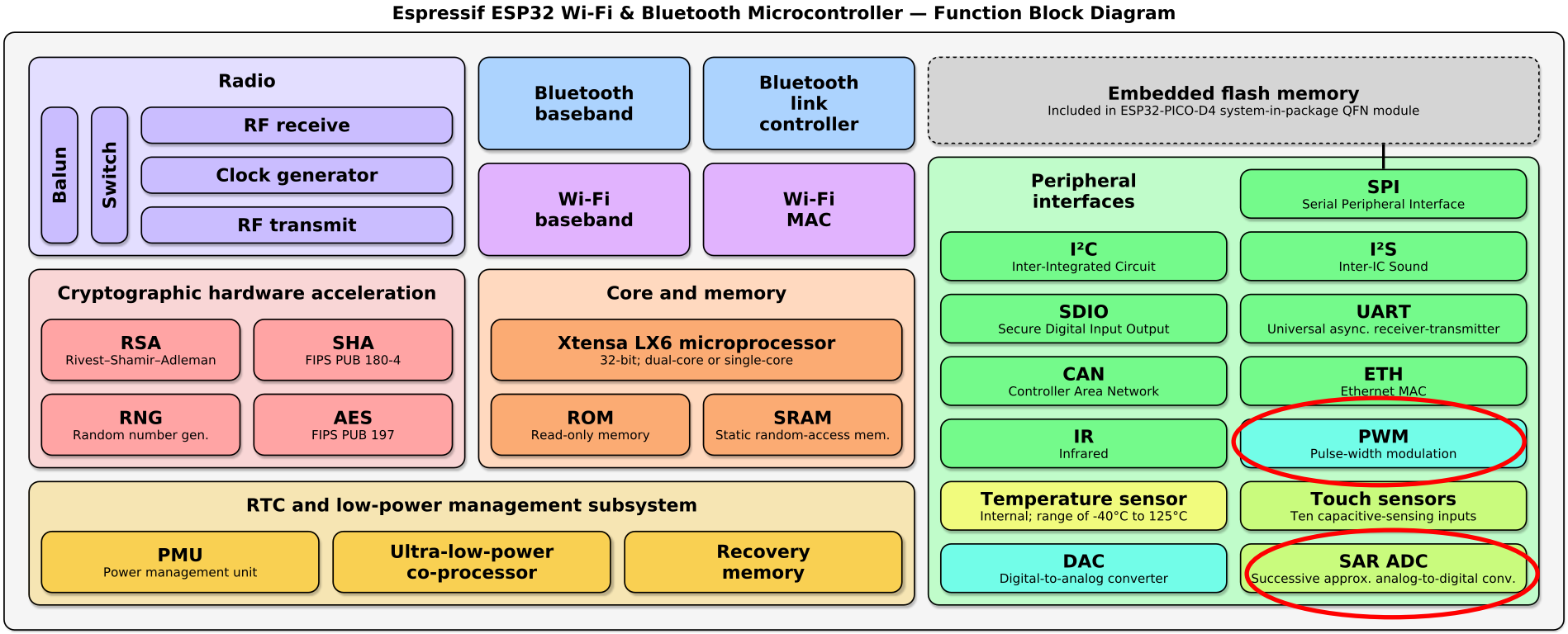
Notes:

- 2/9 project proposal – upload to bcourses by Tues 5 pm
- Survey for Cory Courtyard

Topics

- Checkpoints 1,2,3
- Memory Usage build/runtime
- Updated Skeleton code tour
- Peripheral interface: A/D, pulse count (intro)
- Sensor: line camera

ESP32 Hardware block diagram-peripherals



+GPIO
+ (Ch 18 T.R.M. Pulse Counter)

Sensors

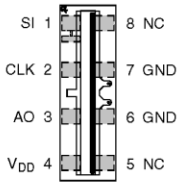
TAOS TEXAS ADVANCED OPTOELECTRONIC SOLUTIONS®

TSL1401CL
128 × 1 LINEAR SENSOR ARRAY WITH HOLD

TAOS136 – JULY 2011

- 128 × 1 Sensor-Element Organization
- 400 Dots-Per-Inch (DPI) Sensor Pitch
- High Linearity and Uniformity
- Wide Dynamic Range . . . 4000:1 (72 dB)
- Output Referenced to Ground
- Low Image Lag . . . 0.5% Typ
- Operation to 8 MHz
- Single 3-V to 5-V Supply
- Rail-to-Rail Output Swing (AO)
- No External Load Resistor Required
- Replacement for TSL1401R–LF
- RoHS Compliant

CL PACKAGE
(TOP VIEW)



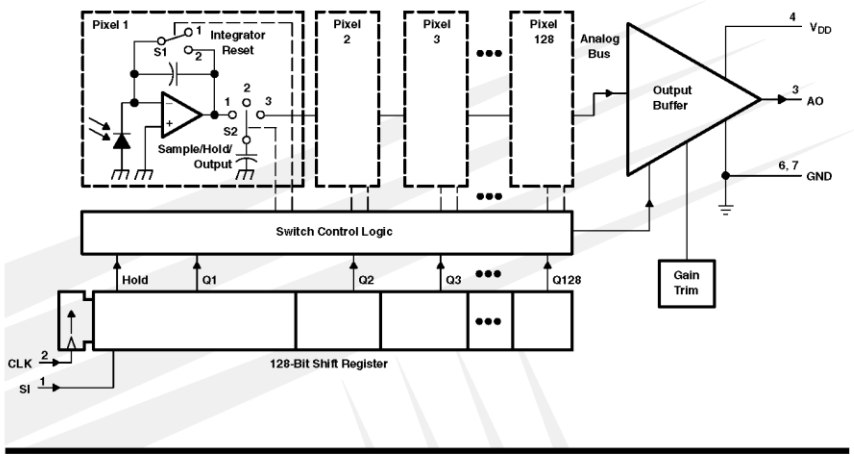
NC – No internal connection

Package Drawing is Not to Scale

Description

The TSL1401CL linear sensor array consists of a 128 × 1 array of photodiodes, associated charge amplifier circuitry, and an internal pixel data-hold function that provides simultaneous-integration start and stop times for all pixels. The array is made up of 128 pixels, each of which has a photo-sensitive area of 3,524.3 square micrometers. There is 8-µm spacing between pixels. Operation is simplified by internal control logic that requires only a serial-input (SI) signal and a clock.

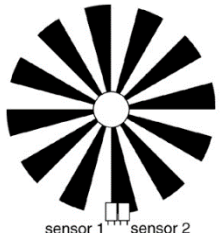
Functional Block Diagram



Line camera: 128 pixels, 200 Hz

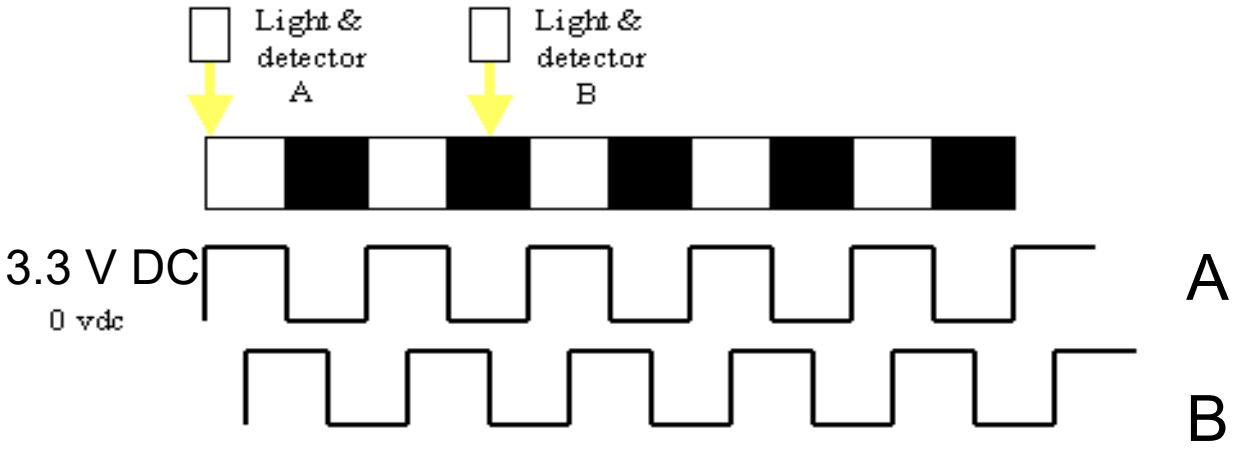
https://www.sparkfun.com/tutorials/283

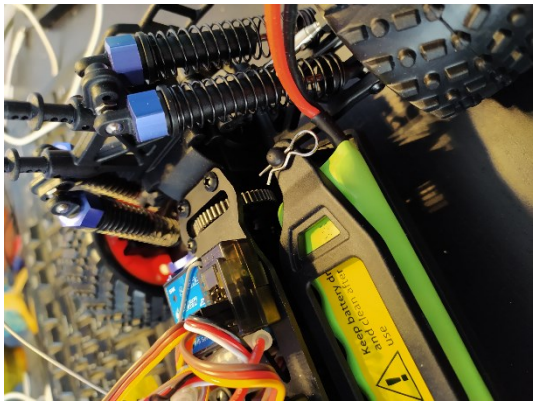
Encoder velocity sensor



Other options? Gyro sensor?

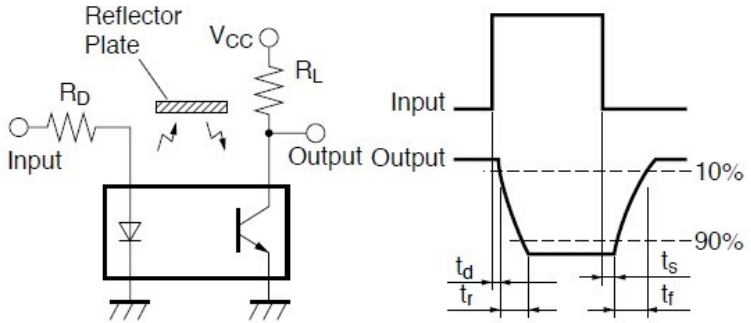
Velocity sensor mounting (preview- week 4)

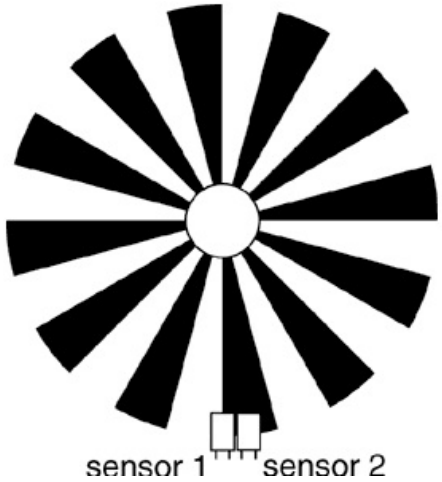




https://www.sinotech.com/wp-content/uploads/quadrature-encoder.gif

Fig.9 Test Circuit for Response Time





100 us response time

? Analog or pulse count input?

ADC Converter (Ch. 30)

```
adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,ADC_ATTEN_DB_0);
int val = adc1_get_raw(ADC1_CHANNEL_0);
```

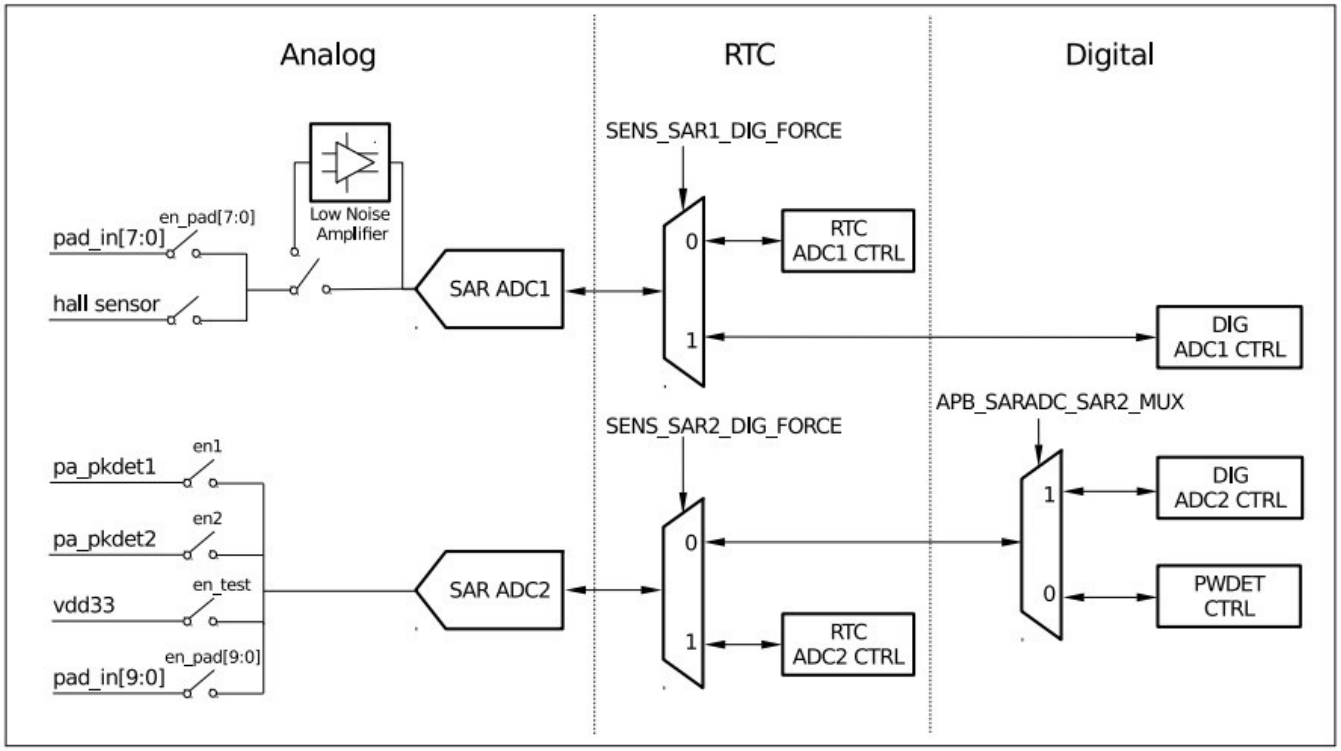


Figure 144: SAR ADC Outline of Function

https://github.com/espressif/esp-idf/tree/release/v4.2/examples/peripherals/adc

Claim: 2 M samples per second (Msps) = 0.5 us

Pulse Counting (Ch 18)

github.com/espressif/esp-idf/tree/release/v4.2/examples/peripherals/pcnt

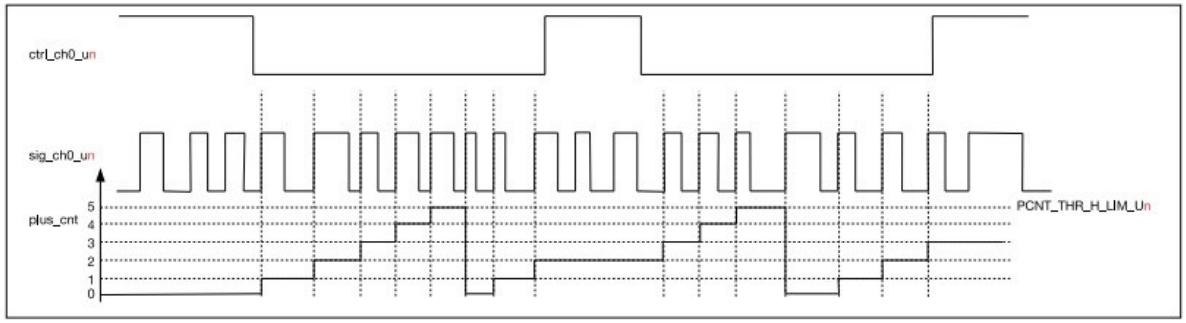


Figure 122: PULSE_CNT Upcounting Diagram

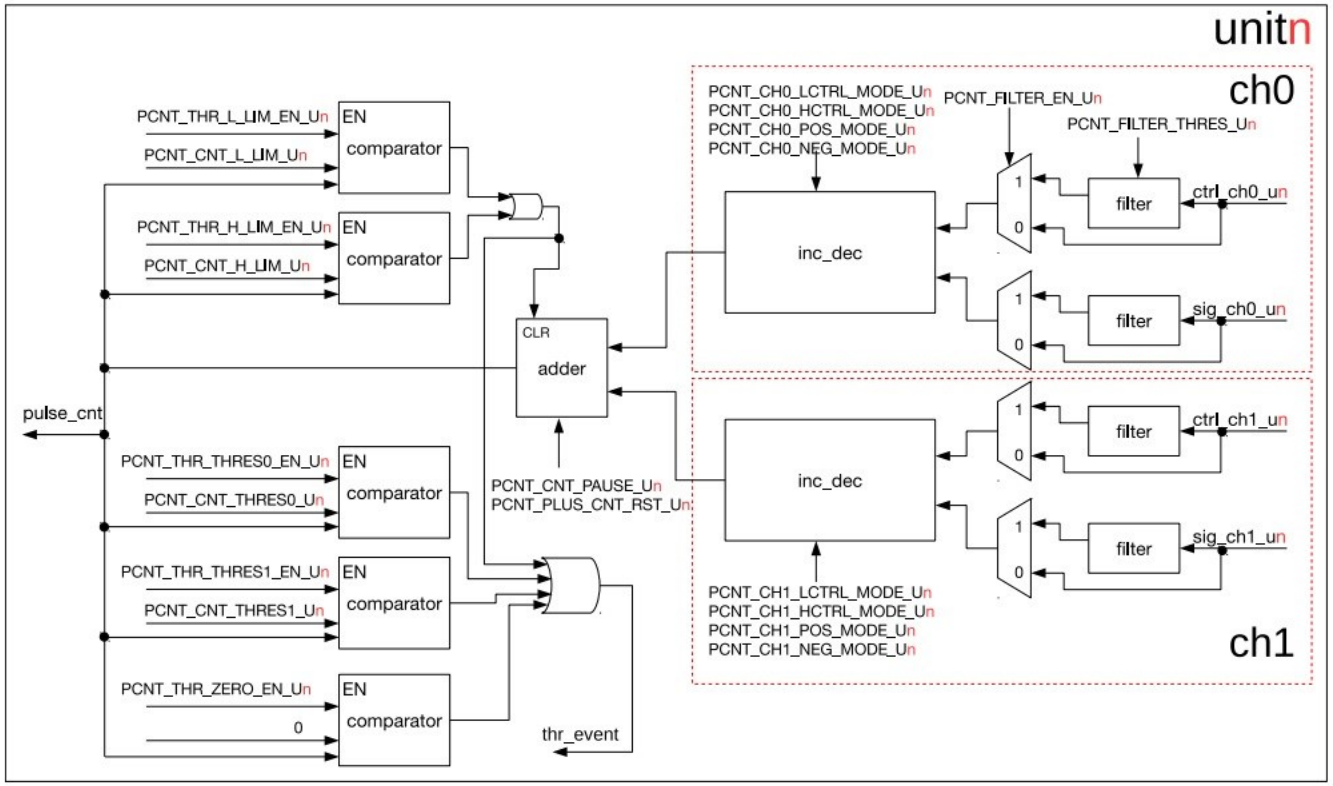
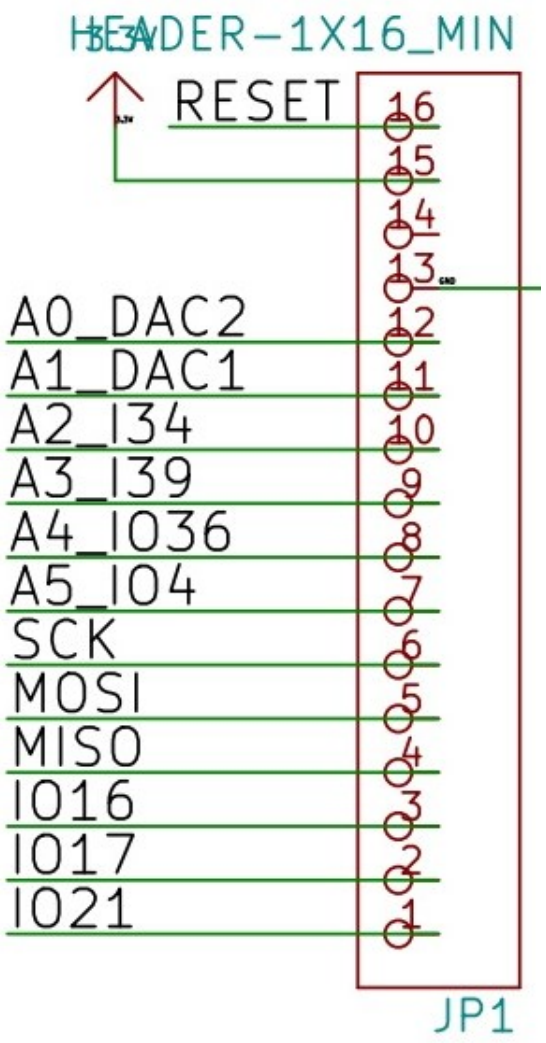
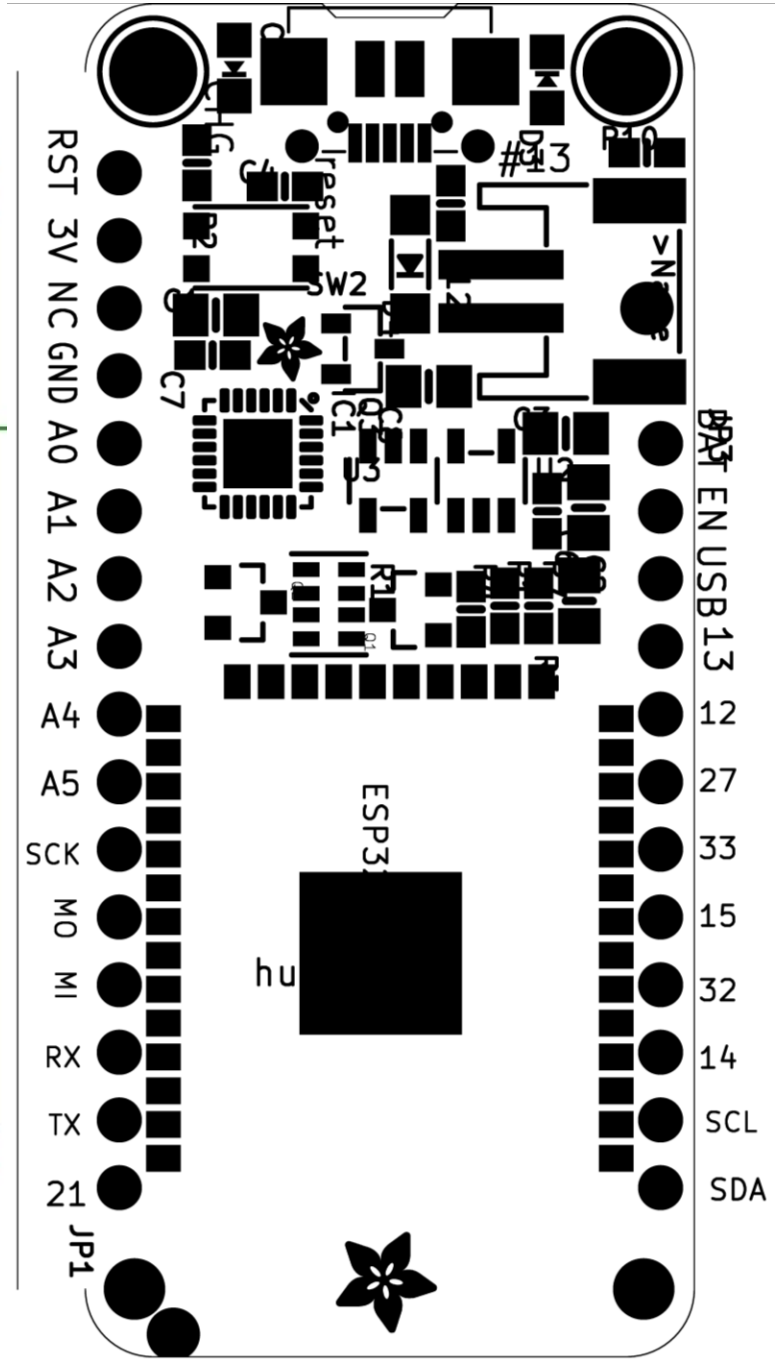
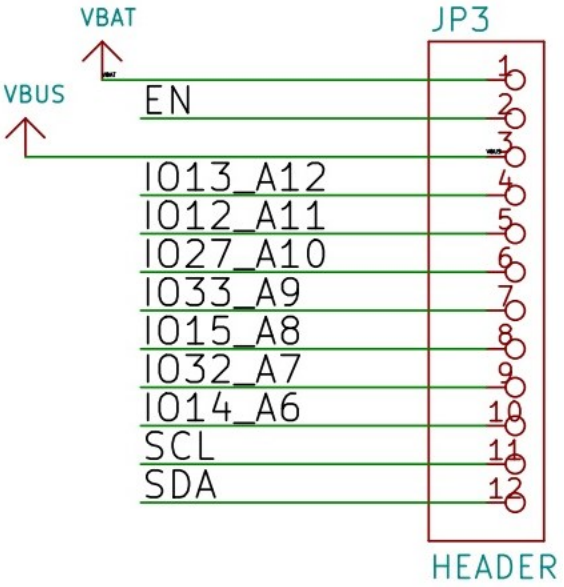


Figure 121: PULSE_CNT Architecture

Huzzah32 Pinouts







ESP32-WROOM Module Connections

Huzzah32

JP1-8
JP1-9
JP1-10
Vref?

JP3-9

JP3-7

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8

P=power
I= input only
I/O = either

ESP32-WROOM Module Connections

Huzzah32

	Name	No.	Type	Function
JP1-11	IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
JP1-12	IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
JP3-6	IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
JP3-10	IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
JP3-5	IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
	GND	15	P	Ground
JP3-4	IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SPI Flash	SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
	SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
	SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
	SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
	SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
	SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS

P=power
I= input only
I/O = either

ESP32-WROOM Module Connections

Huzzah32

JP3-8

RTS?
DTR?

JP1-7
JP1-3
JP1-2
JP1-6
JP1-5
JP1-4

JP1-1

UART

JP3-11
JP3-12

IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

P=power
I= input only
I/O = either

EECS192 Lecture 3
Feb. 2, 2021

Notes:

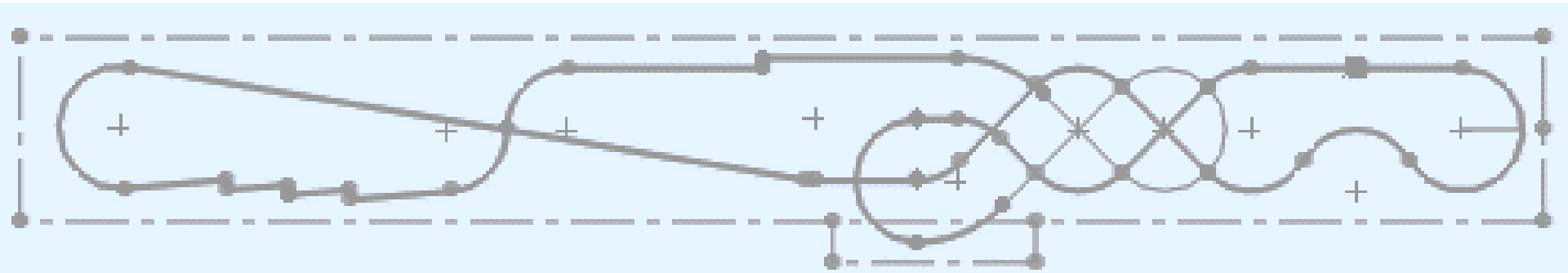
- 2/9 project proposal – upload to bcourses by Tues 5 pm
- Survey for Cory Courtyard

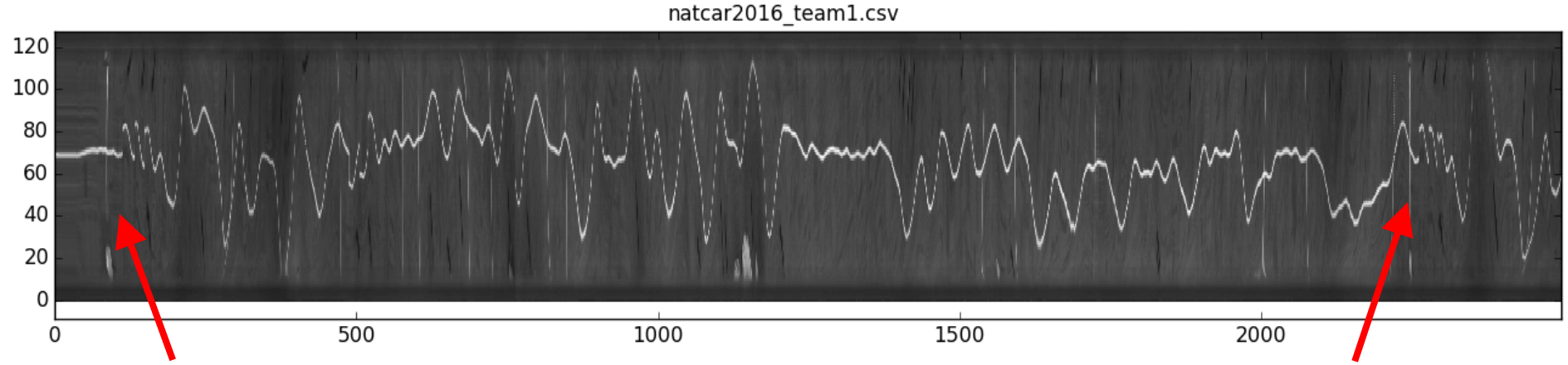
Topics

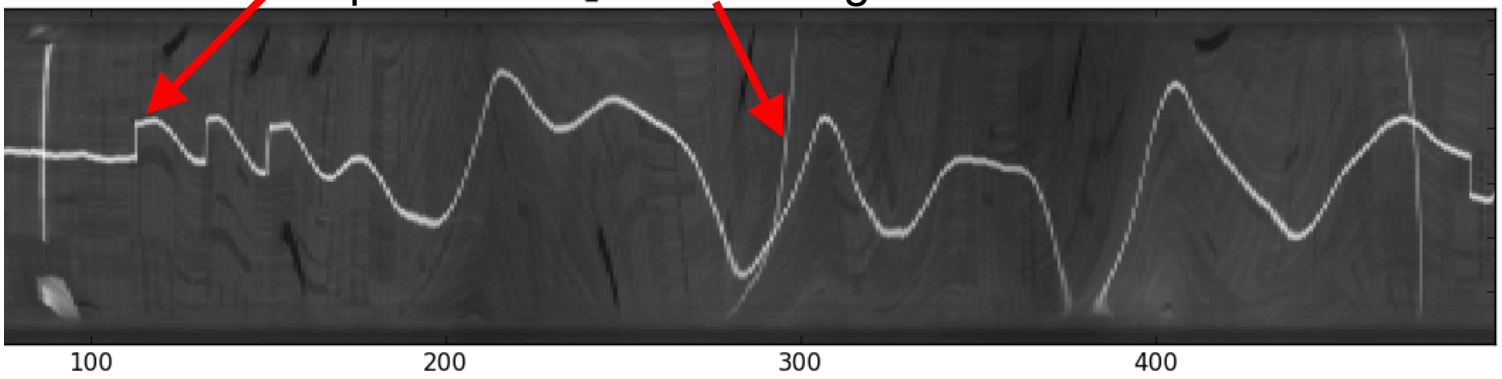
- Checkpoints 1,2,3
- Memory Usage build/runtime
- Updated Skeleton code tour
- Peripheral interface: A/D, pulse count (intro)
- Sensor: line camera

Line Camera Output

Example track- Cory 3rd floor

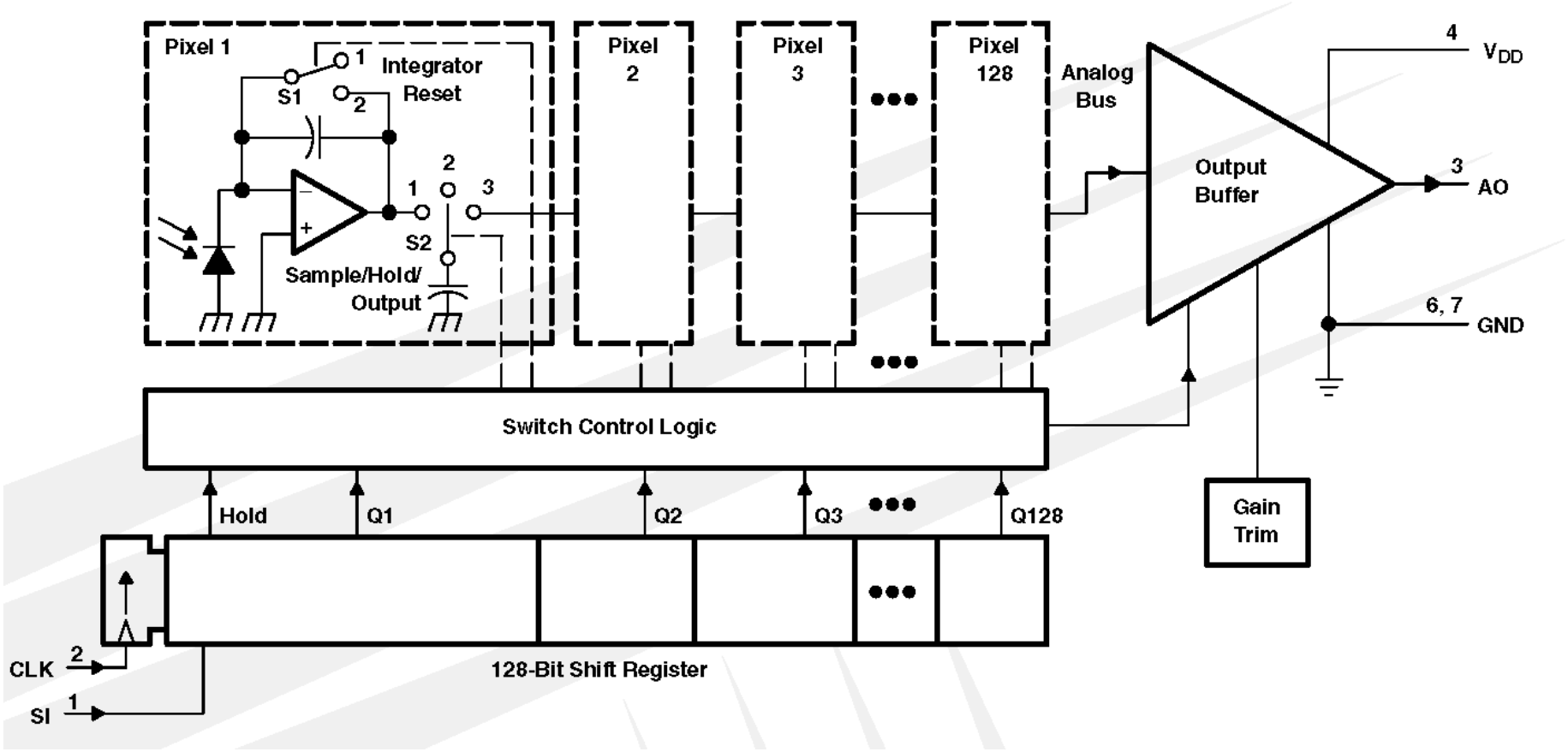






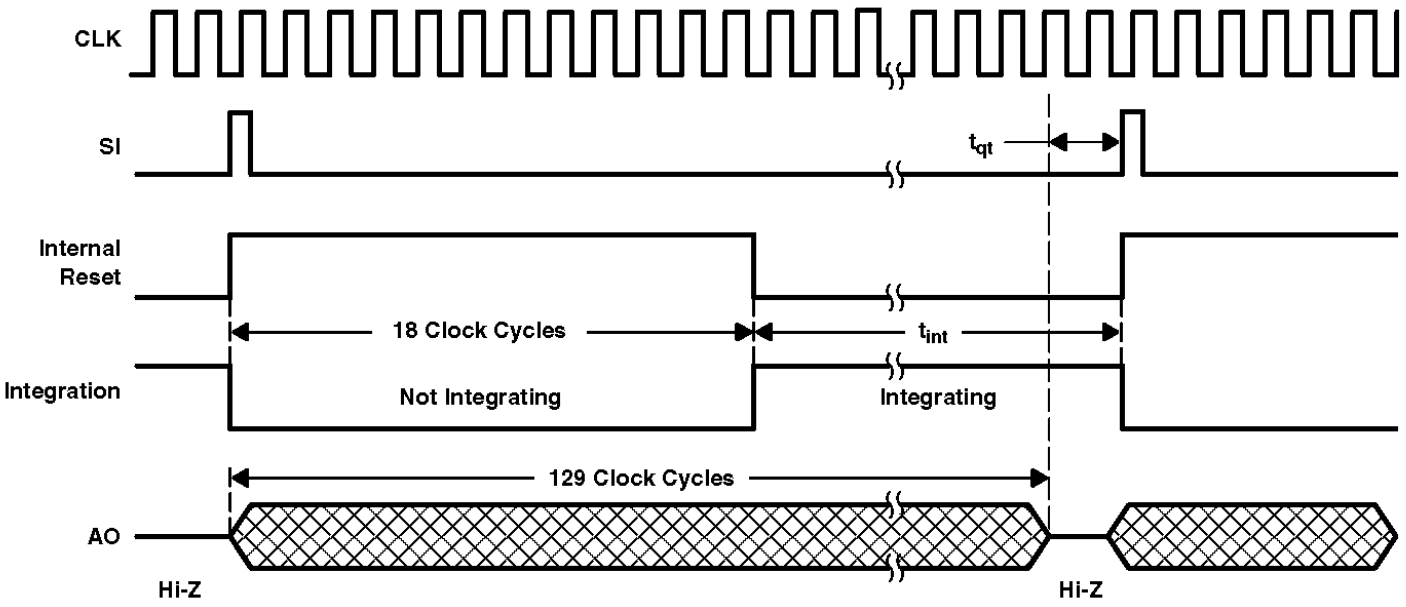
TSL 1401 line sensor

Functional Block Diagram



SI: serial input. SI defines the start of the data-out sequence.
AO: analog output
CLK: clock

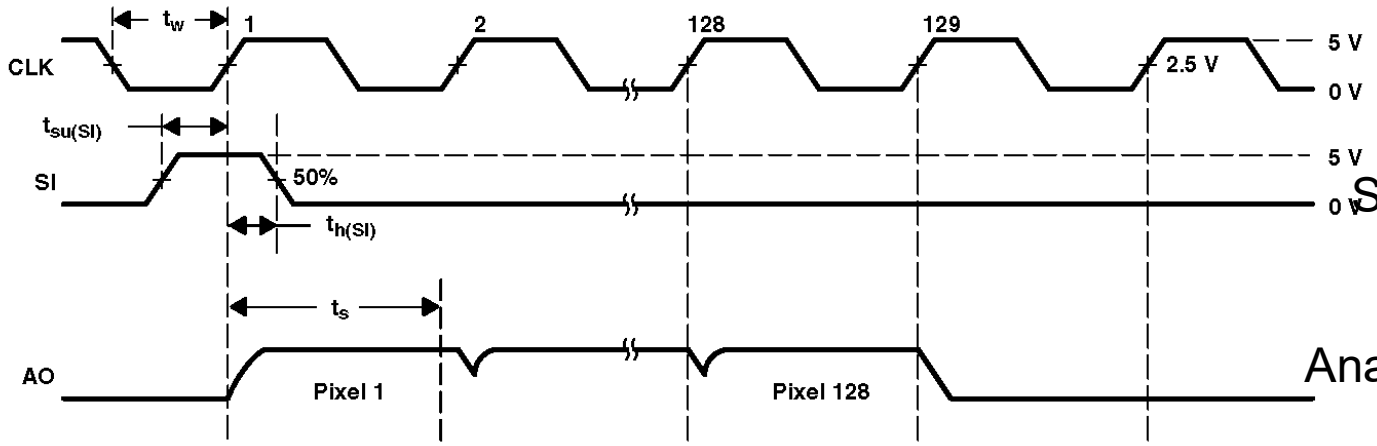
TSL 1401 line sensor



From PWM?

From PWM or GPIO?

Figure 1. Timing Waveforms

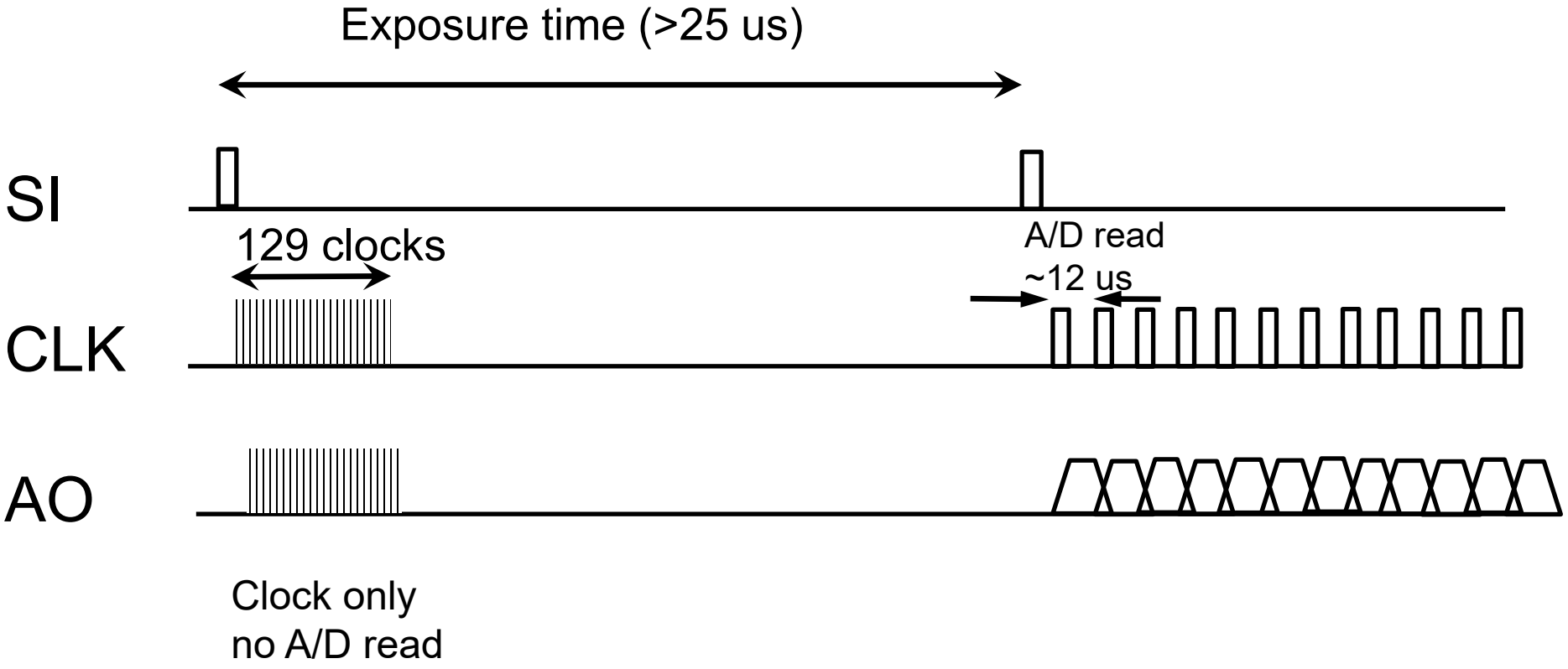


Start integration

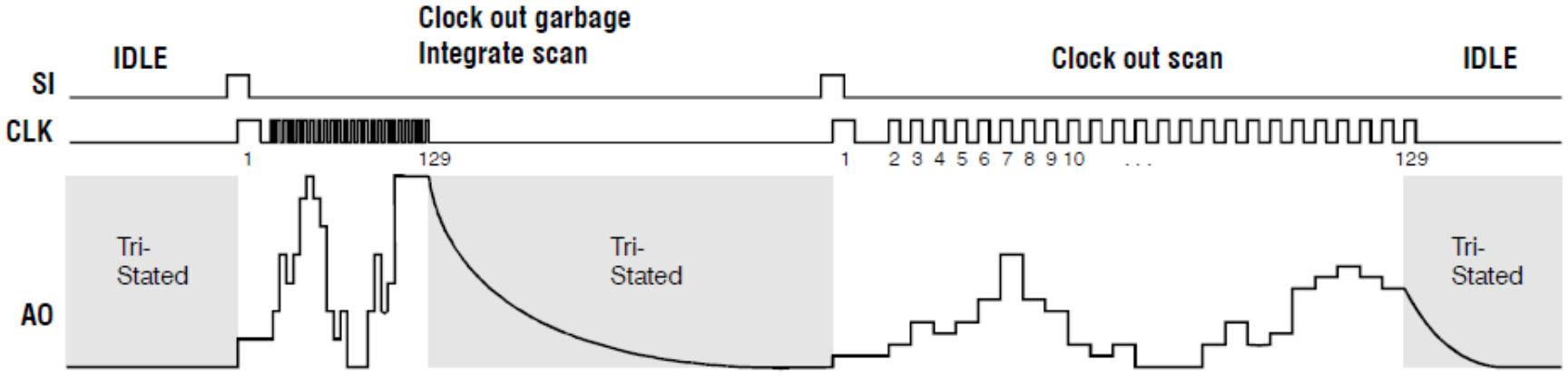
Analog output to A/D

Figure 2. Operational Waveforms

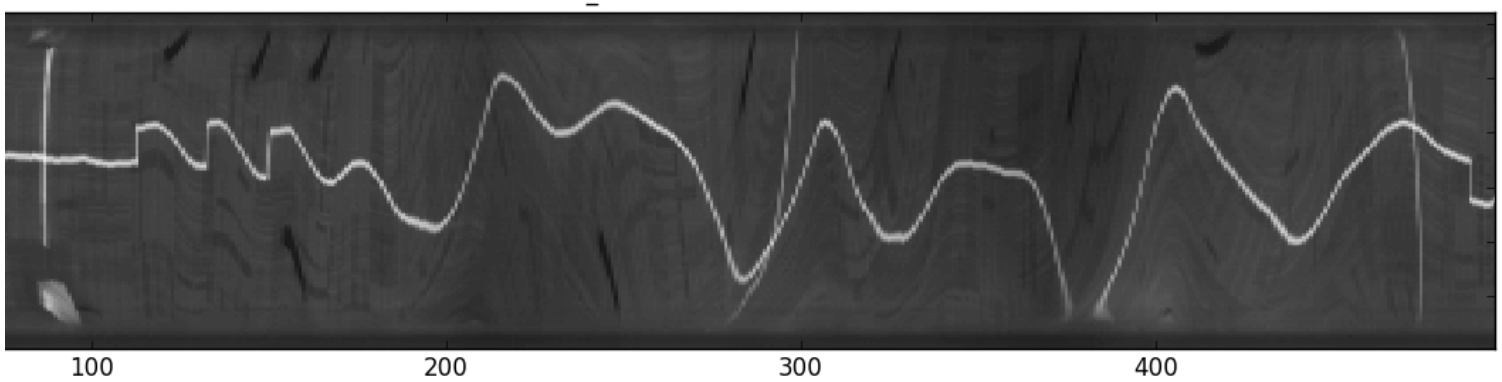
TSL 1401 line sensor- option exposure control



Automatic Gain Control



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



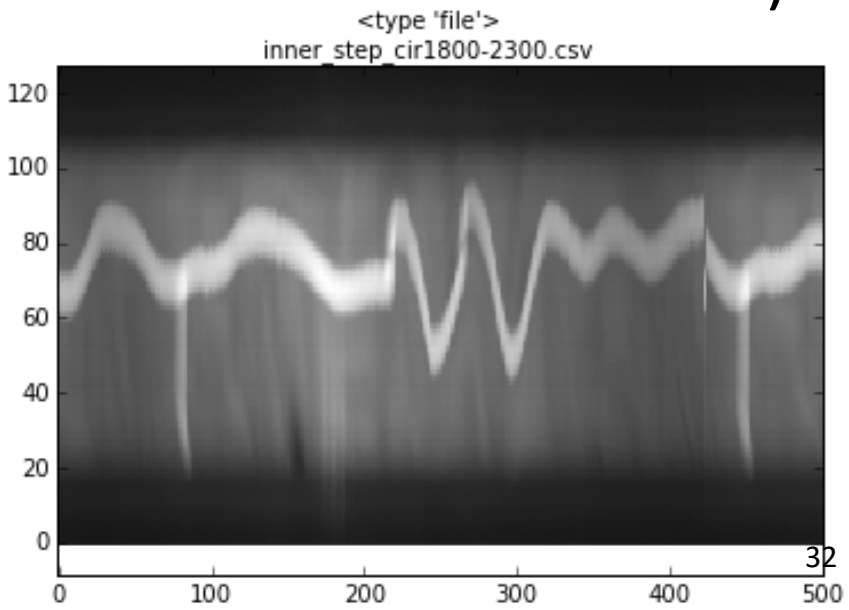
- Choose exposure time based on average illumination
- Keep frame rate constant e.g. read sensor <u>twice 1+4 ➔ 4 +1 ms</u>
- (Constant time is important for control- will see later)

Possible algorithms for line detection

e.g. prototype with scipy.signal.filter. Many options. Here are 3 suggestions:

- 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

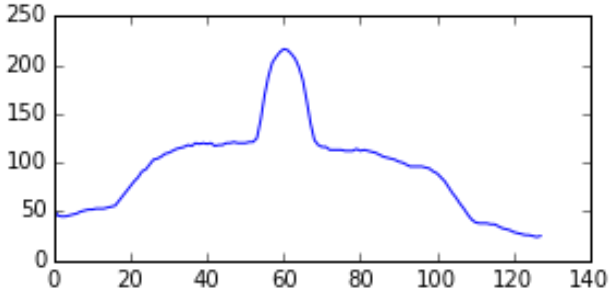
Try out algorithms in HW1



Alternative #1 frame subtraction

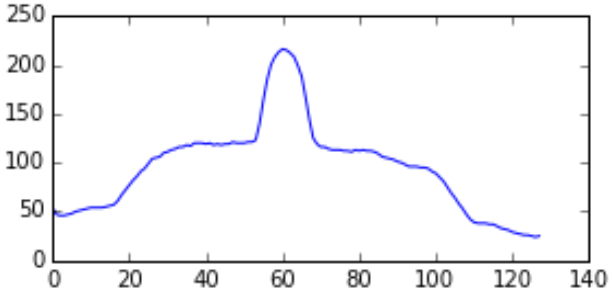
TSL 1401 line sensor 8 bit

Frame 0



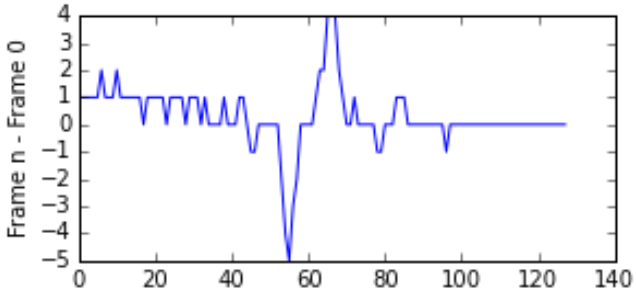
Frame 0

Frame 1



Frame 2

Frame 1-Frame 0



Frame 2-Frame 0

Notes: peak shows edge of track. Noisy, only 1 pixel resolution.

Alternative #2 Difference of Gaussians

Laplacian of Gaussian

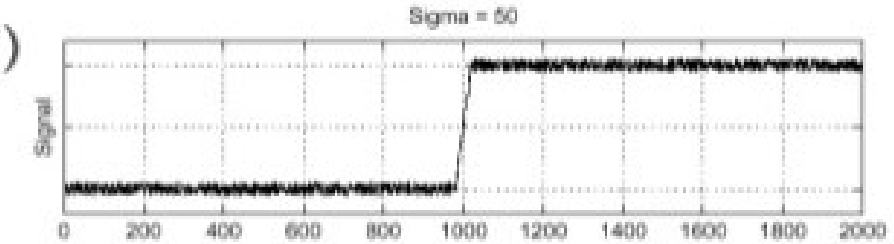
$$\triangle[G_\sigma(x,y) * f(x,y)] = [\triangle G_\sigma(x,y)] * f(x,y) = LoG * f(x,y)$$

Convolve with Difference of Gaussians kernel (approx. to LoG)

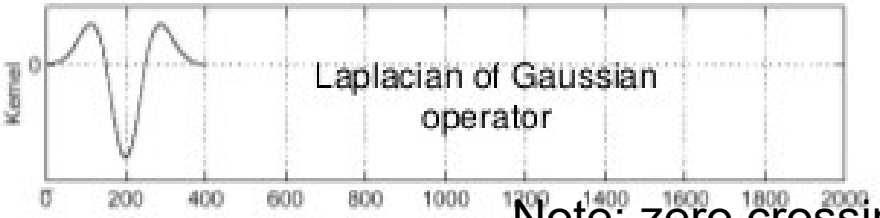
$$\Gamma_{\sigma_1,\sigma_2}(x) = I * \frac{1}{\sigma_1\sqrt{2\pi}} e^{-(x^2)/(2\sigma_1^2)} - I * \frac{1}{\sigma_2\sqrt{2\pi}} e^{-(x^2)/(2\sigma_2^2)}.$$

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

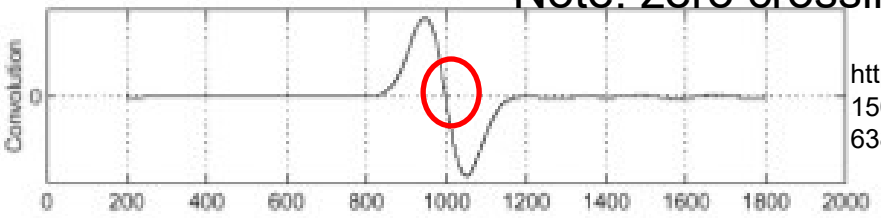
f



$\frac{\partial^2}{\partial x^2}h$



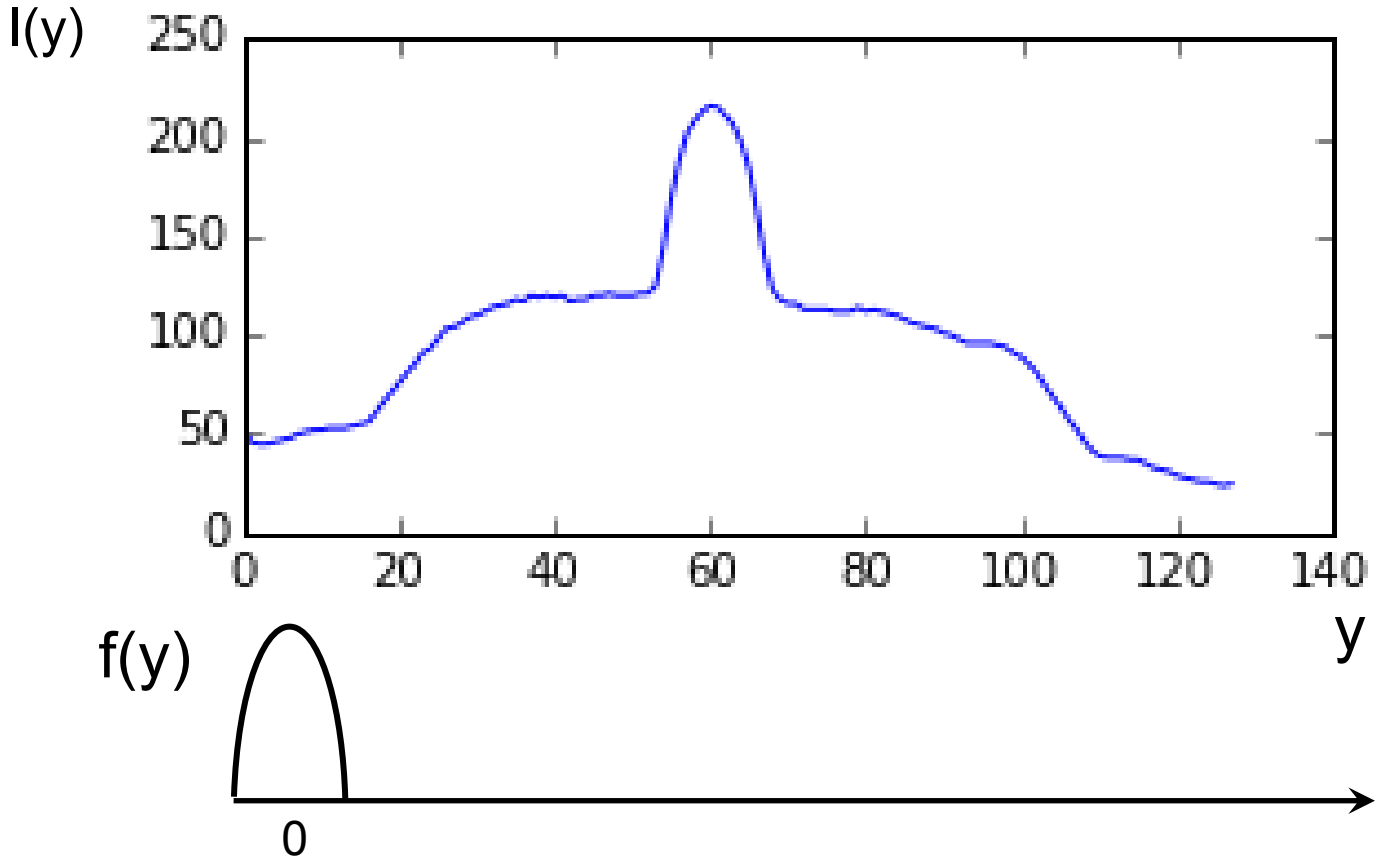
$(\frac{\partial^2}{\partial x^2}h) \star f$



Note: zero crossing is edge location

https://image.slidesharecdn.com/cbirfeatures-150705141111-lva1-app6892/95/cbir-features-47-638.jpg?cb=1436105787

Alternative #3 Correlation



$$\arg\min_{\Delta y} \; \| I(y) - f(y - \Delta y) \|_2$$

Notes: normalize, find by least squares or search. Can use $\Delta y(n-1)$ to initialize

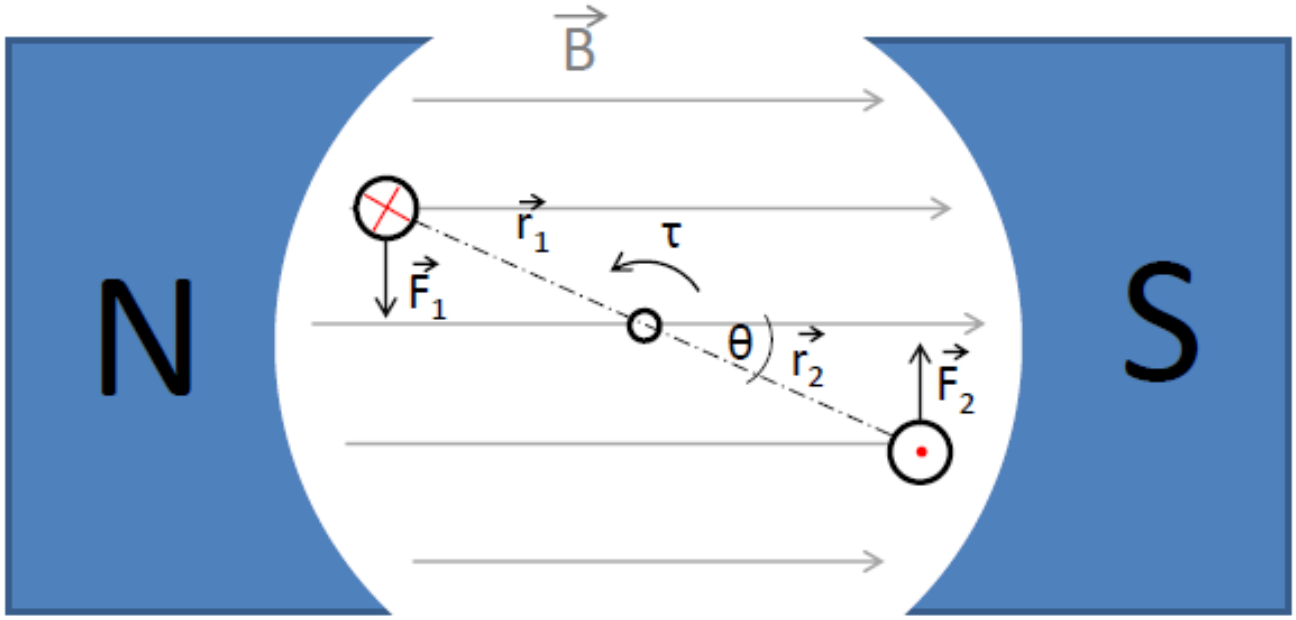
Summary

Memory Usage build/runtime
Updated Skeleton code tour
Peripheral interface: A/D, pulse count (intro)
Sensor: line sensor

Keep in mind limited time and memory

Extra Slides

DC Motor Physical Model

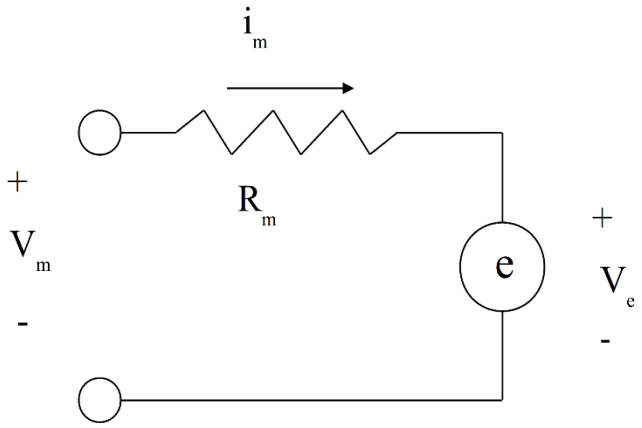


$$\vec{F} = i\vec{l} \times \vec{B}$$

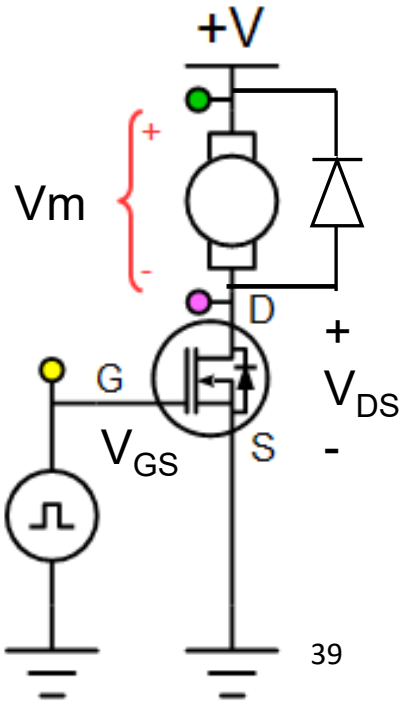
$$\tau = \vec{r}_1 \times \vec{F}_1 + \vec{r}_2 \times \vec{F}_2$$

Motor Electrical Model (neglect inductor)

Motor Electrical Model
Back EMF
Motor electromechanical behavior



Continued on board
Also- see motor worksheet……

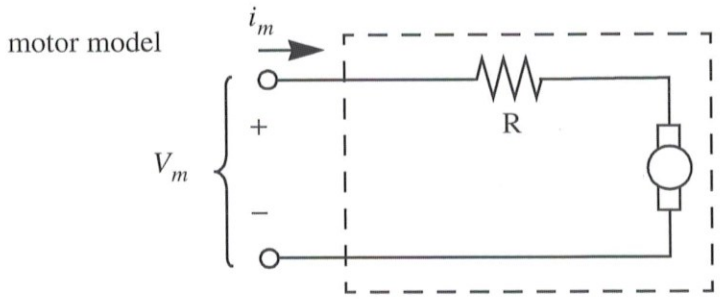


Note: missing e-stop!

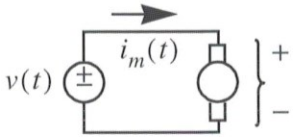
Motor model

For this problem, consider a DC permanent magnet motor (as used in your car). The car is on a carpet and moves in a straight line with no slip between the wheels and the carpet. The car is initially moving at a speed of 2 meters per second.

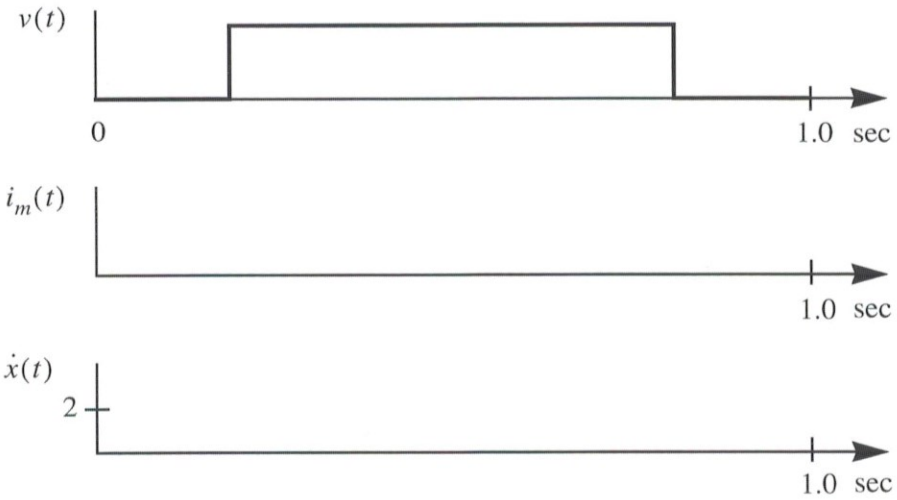
You can assume a motor model as shown below. The qualitative shape of the curves is more important than magnitudes.



[4 pts.] **a)** Consider the motor driven from a voltage source with voltage $v(t)$, as shown. Sketch car velocity $\dot{x}(t)$ and motor terminal current for the time indicated.

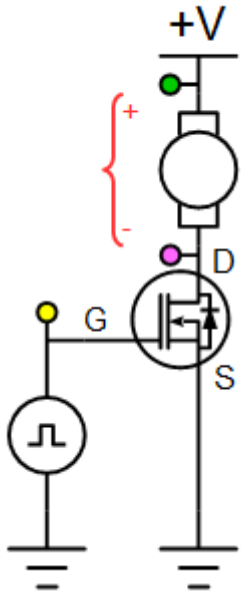


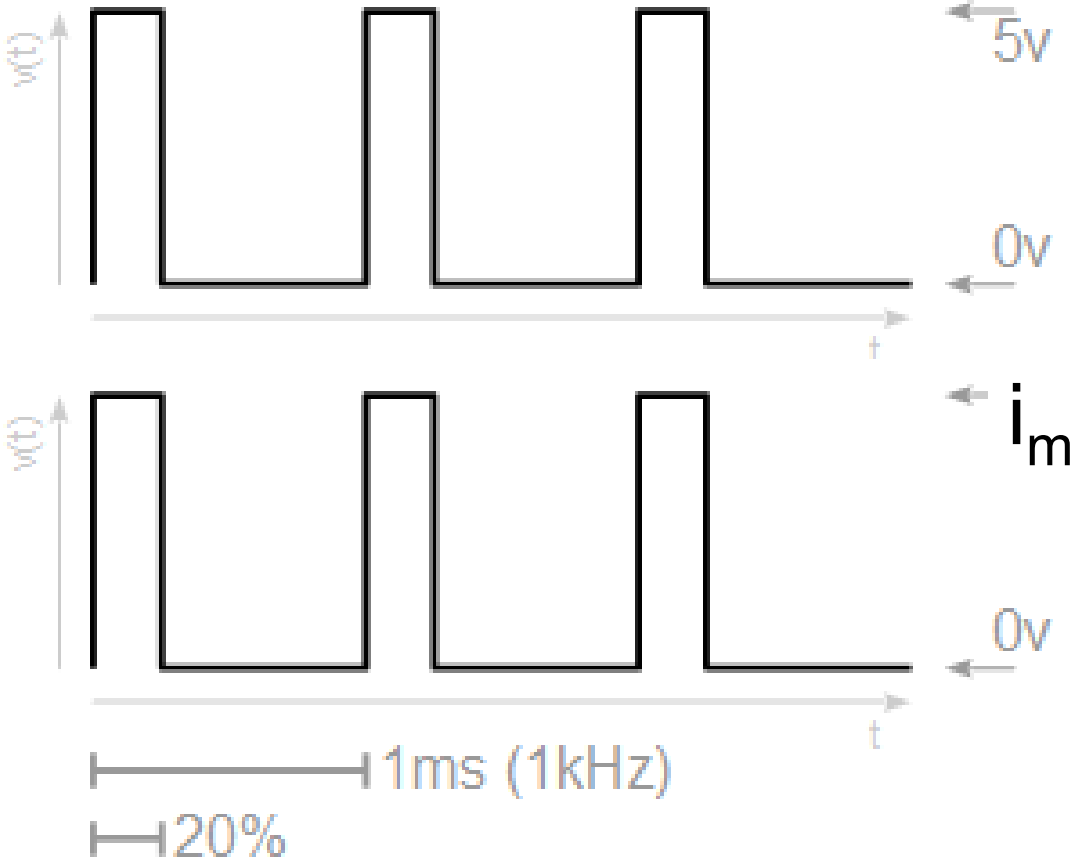
Let peak speed = 5 m/sec
Accel = 5 m/s^2
k_e = 1 v/(m/sec)
On board



(for answer
see sp99 final solution)

PWM for Main Motor control



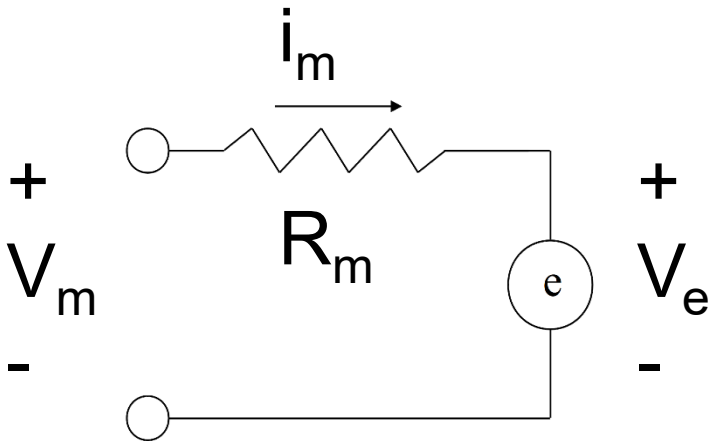


$\langle i_m \rangle = (T/T_o)\, i_{max}$

Is i_{max} constant?

Motor Electrical Model

Motor Electrical Model
Back EMF
Motor electromechanical behavior

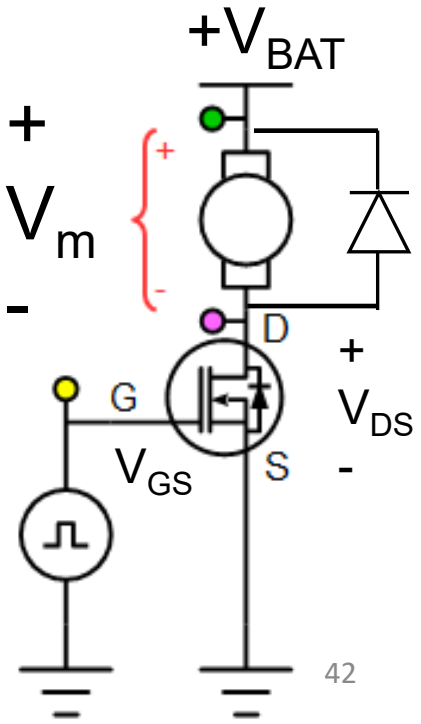


Also- see motor worksheet……

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

Conclusion:
$\langle i_m \rangle$ =?

Motor Resistance?
Peak current?



Motor Model

On board….

http://inst.eecs.berkeley.edu/~ee192/sp18/files/NiseAppendixI.pdf

http://inst.eecs.berkeley.edu/~ee192/sp13/pdf/motor_modeling.pdf