EECS192 Lecture 7
Motor Modelling and Steering Introduction
Mar. 2, 2021

Topics

- Checkpoint 5: feedback
- Checkpoint 6: closed-loop control
- Time spent survey
- HW 1 Notes
- PWM for motor drive/MOSFET intro
- Steering control II
- Embedded Issues: deadlock

Checkpoint 5 feedback

1. printf delay destabilizing control, and disrupting velocity estimate
    1. Use snprintf() then log_add() (either UDP or UART)

2. Tasks: WDT

3. Resource sharing Timer, Core0 and Core1

Monitoring FreeRTOS Tasks- IDLE0/1

```
void print_tasks();
```

# of tasks 12

Task name	**number of cycles**		
IDLE0	283440623	49%	(CPU 0)
usertask	142791054	24%	
IDLE1	145004887	25%	(CPU 1)
heartbeat	6661	<1%	
timer_evt_task	194739	<1%	
Tmr Svc	55	<1%	
control_task	60360	<1%	
esp_timer	209	<1%	
ipc0	10215	<1%	
main	95764	<1%	
log_task	13490	<1%	
ipc1	15121	<1%	(inter process comm?)

➔ Starving the ``idle" process (will cause a crash -wdt).
➔ Make sure every process has vTaskDelay() for a lower priority process to run

CP6- Closed Loop Track with Velocity Control 3/12

Set up a figure 8* track. Use 1 meter string with chalk attached to make circles, and connect with tangent lines, and 60 degree crossing. Use white masking tape for figure 8 if on light background, or black tape if on light background.

C6.1 Show car driving the figure 8*, at speed of 1 m/sec or better. (May be lower for small circle.)

(You may use a wireless command to tell the car to start or stop running, but no other commands may be sent to the car.)

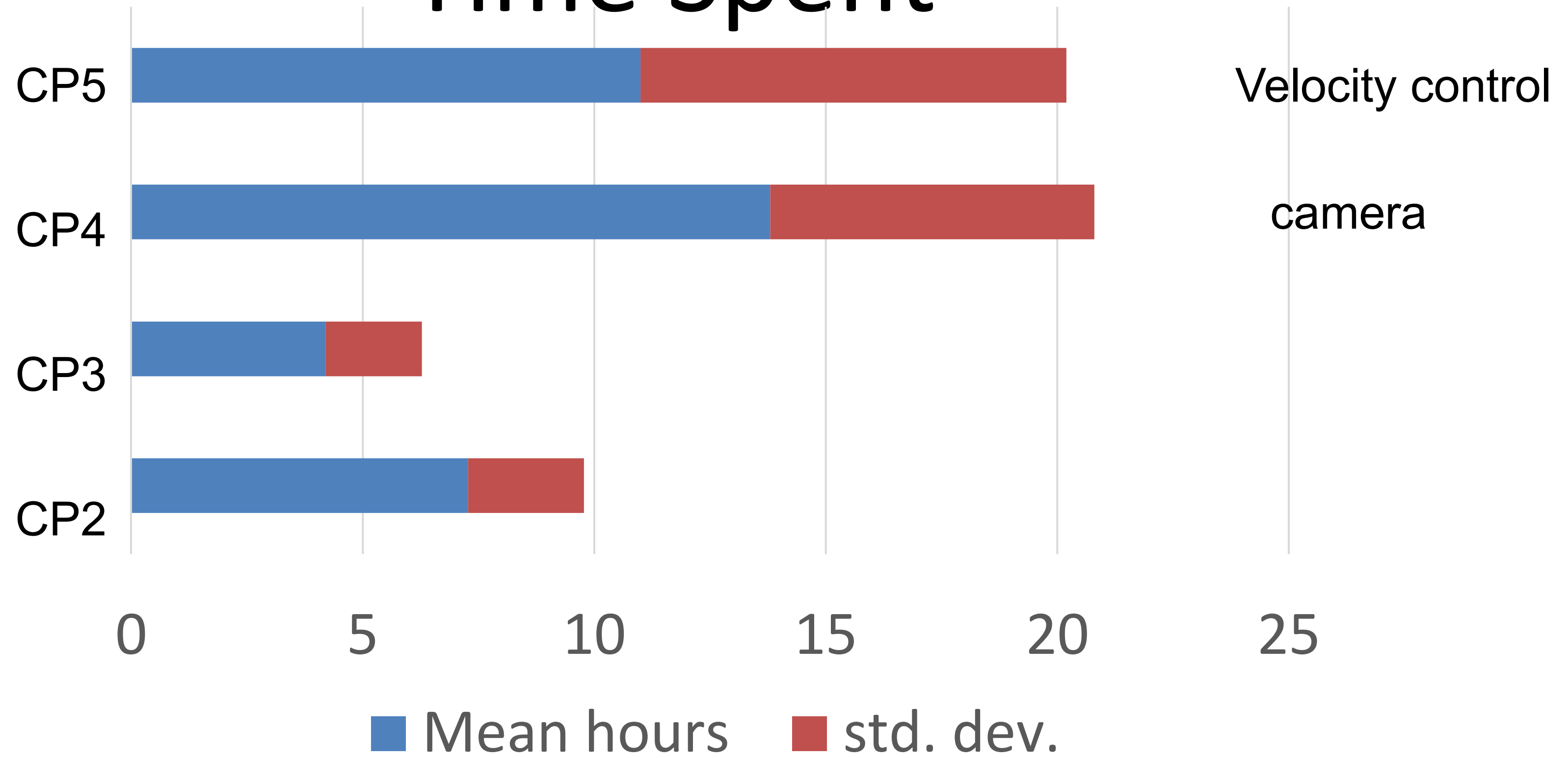
C6.2 Submit plots on one graph: steering angle command (degrees or radians), track error (cm), ESC command (% full speed), sensed velocity, all versus time axis in seconds.

C6.3 All members must fill out the checkpoint survey before the checkoff close. Completion is individually graded.

**** If you do not have space for a full size figure 8, use smaller than 1 m radius to fit. If you do not have room for a figure 8, use a circle of up to 1 m radius.***

(example Amazon tape): https://www.amazon.com/Removable-Painters-Painting-Labeling-Stationery/dp/B082R27TP6/ref=sr_1_7?dchild=1&keywords=1+inch+white+masking+tape&qid=1613947385&s=industrial&sr=1-7

Time Spent



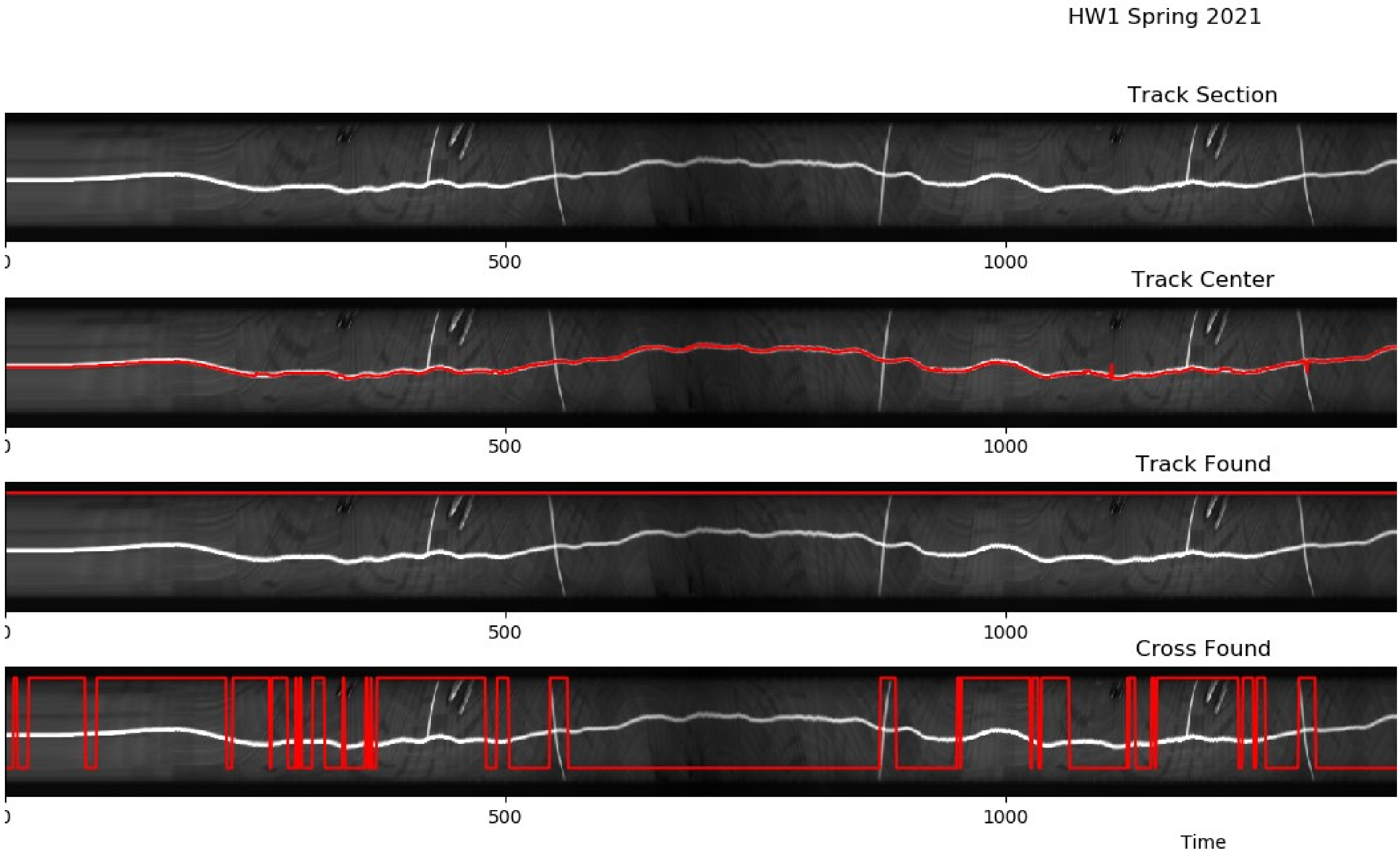
If you are stuck on one thing for more than 4 hours, it is time to ask for help such as Piazza, office hours, or email for extra help.

People have very different backgrounds, so time varies widely
(4-40 hours CP5)

If you are spending more than 10 hours, asking for advice in office hours should be productive rather than figuring everything out by yourself.

HW 1 notes

Crossing Algorithm on test data

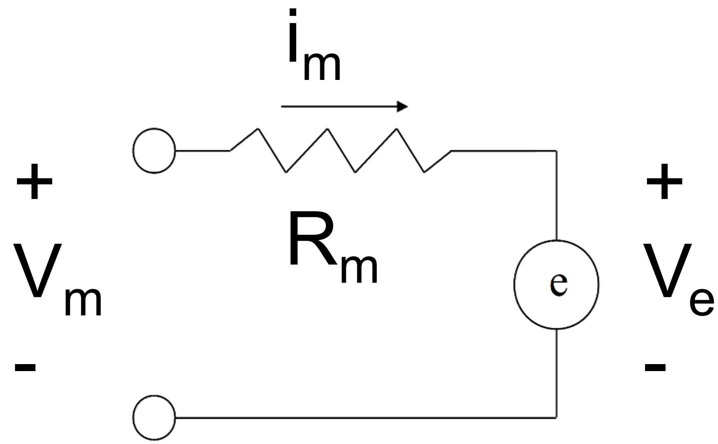


EECS192 Lecture 7
Motor Modelling and Steering Introduction
Mar. 2, 2021

Topics

- Checkpoint 5: feedback
- Checkpoint 6: closed-loop control
- Time spent survey
- HW 1 Notes
- PWM for motor drive/MOSFET intro
- Steering control II
- Embedded Issues: deadlock

Motor Electrical Model



Motor Electrical Model
Back EMF
Motor electromechanical behavior

Also- see motor worksheet……
https://inst.eecs.berkeley.edu/~ee192/sp21/docs/motor-worksheet.pdf

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

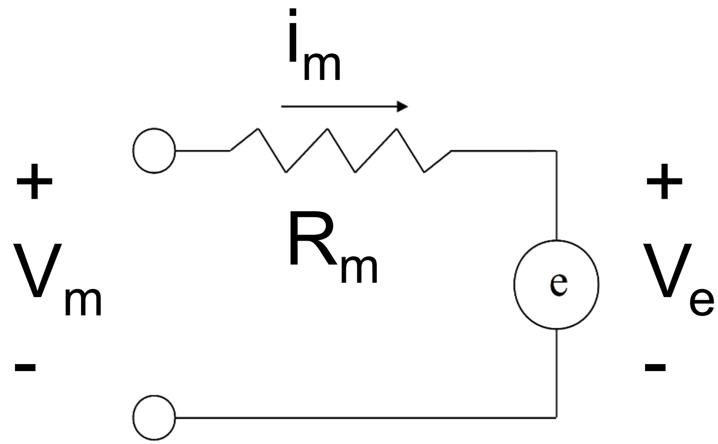
Torque equation: $\tau = k_\tau \, i_m$

Back EMF equation: $V_e = k_e \dot{\theta}_m$

Conclusion:
$\langle i_m \rangle = ?$

Motor Resistance?
Peak current?

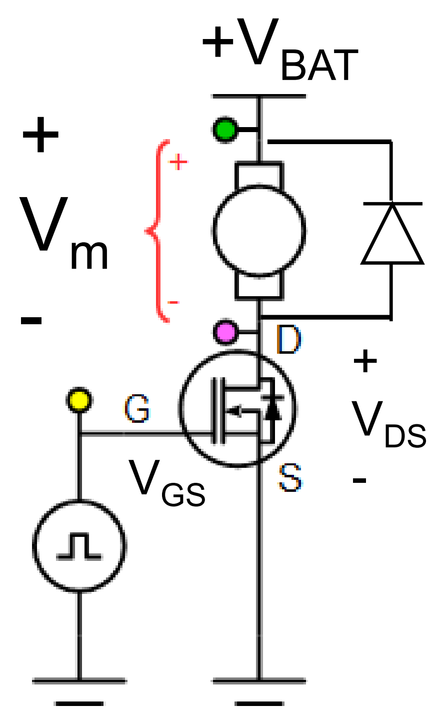
Motor Electrical Model



Motor Electrical Model
Back EMF
Motor electromechanical behavior

Also- see motor worksheet……

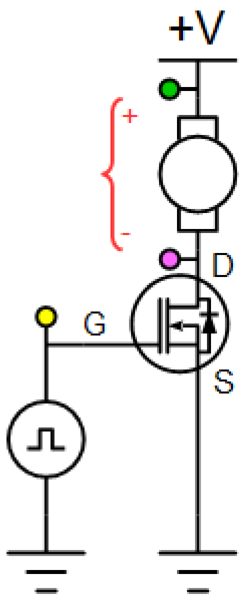
$$i_m = \frac{V_{BAT} - k_e \dot{\theta}_m}{R_m}$$

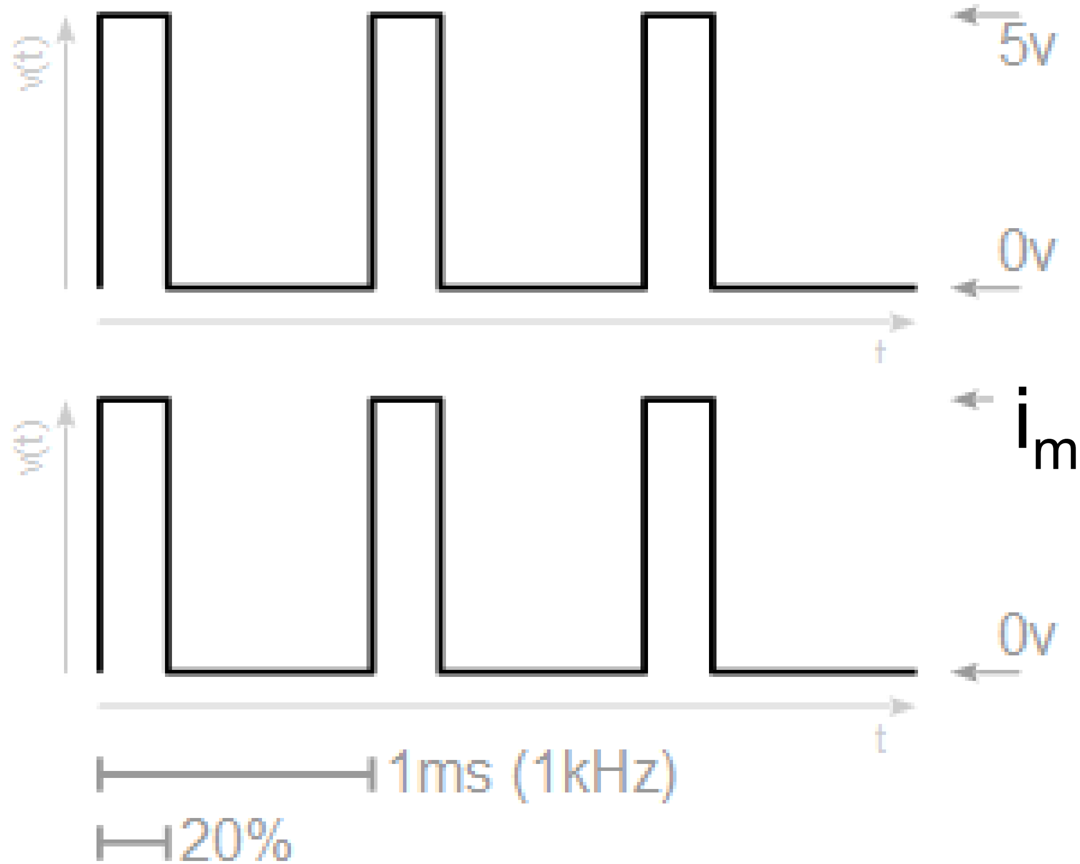


Motor Resistance?
Peak current?

Conclusion:
$\langle i_m \rangle = ?$

PWM for Main Motor control

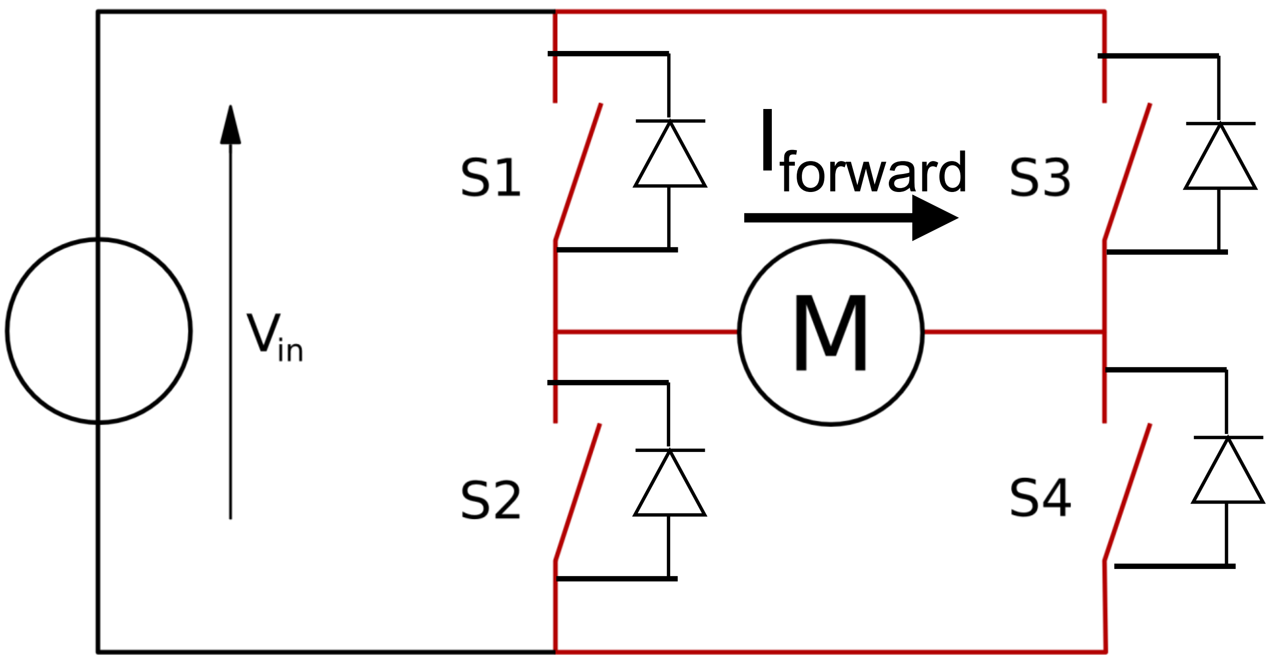




$$\langle i_m \rangle = (T/T_o)\, i_{max}$$

Is i_{max} constant?

H Bridge Concept



S1	S2	S3	S3	Function?
Off	Off	Off	Off	
On	Off	Off	On	
Off	On	On	Off	
On	On	Off	Off	
On	Off	On	off	
Off	On	Off	on	

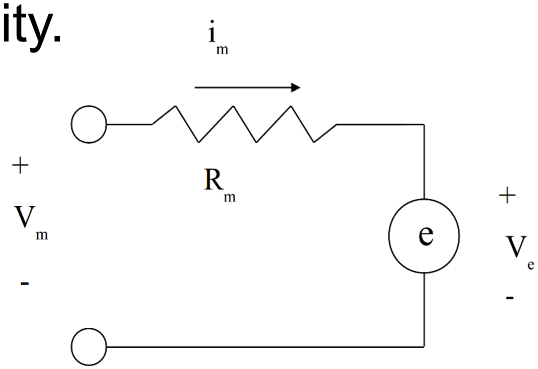
Practice Q2 (3/9)

https://inst.eecs.berkeley.edu/~ee192/sp21/files/motor-worksheet-soln.pdf

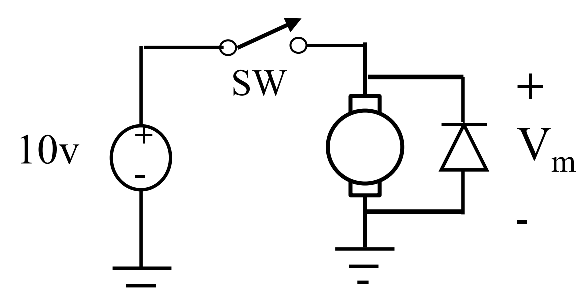
Consider a DC permanent magnet motor (as used in your car).
The car is initially at rest. The motor is connected as shown below.
Neglect battery and switch resistance. Neglect motor inductance. Assume diode is ideal.
Assume motor resistance = 0.2 ohm, and that the car accelerates to 4 m/s in 2 seconds.
Assume back EMF constant is 1V/(m/s).
Assume time constant for deceleration is 1 second.

Switch turns on at 0 sec, off at 2 sec.

Complete the sketches below for motor current i_m, motor voltage V_m, and car velocity.



Motor model



Motor connection

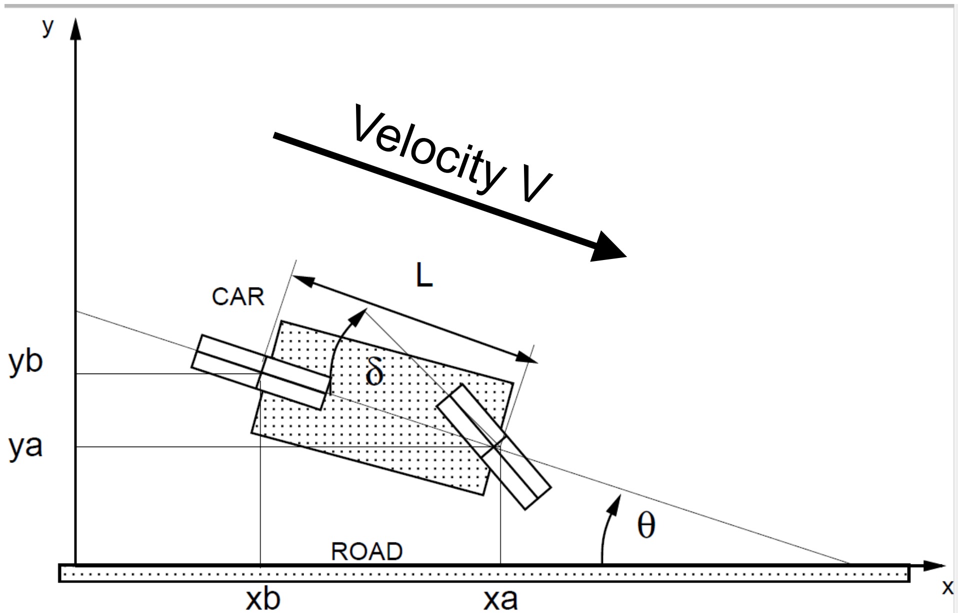
EECS192 Lecture 7
Motor Modelling and Steering Introduction
Mar. 1, 2021

Topics

- Checkpoint 5: feedback
- Checkpoint 6: closed-loop control
- Time spent survey
- HW 1 Notes
- PWM for motor drive/MOSFET intro
- Steering control II
- Embedded Issues: deadlock

Bicycle Steering Model

More detailed models see: https://inst.eecs.berkeley.edu/~ee192/sp15/refSteer.html



$$\dot{x}_b = V cos(\theta(t)) \quad (1)$$

$$\dot{y}_b = -V sin(\theta(t)) \quad (2)$$

$$\dot{\theta} = \frac{V}{L} tan(\delta(t)) \quad (3)$$

$$y_a = y_b - L sin(\theta(t)) \quad (4)$$

Bicycle Steering Model-linearized

$$\dot{x}_b = V cos(\theta(t))$$

$$\dot{y}_b = -V sin(\theta(t))$$

$$\dot{\theta} = \frac{V}{L} tan(\delta(t))$$

$$y_a = y_b - L sin(\theta(t))$$

Original non-linear equations

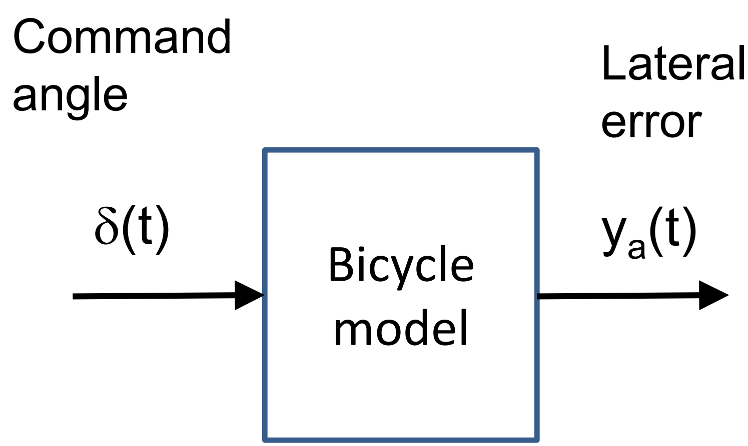
<u>Assume small angle, constant V:</u>

$$\dot{y}_b \approx -V\theta$$

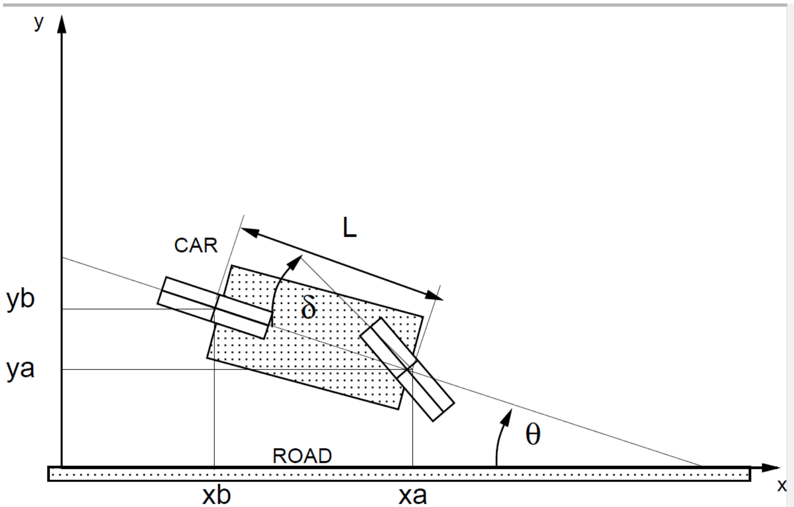
$$\dot{\theta} \approx \frac{V}{L}\delta(t)$$

$$\dot{y}_a \approx \dot{y}_b - L\dot{\theta} = -V\theta - L\dot{\theta}$$

$$\ddot{y}_a = \frac{-V^2}{L}\delta(t) - V\dot{\delta}(t).$$

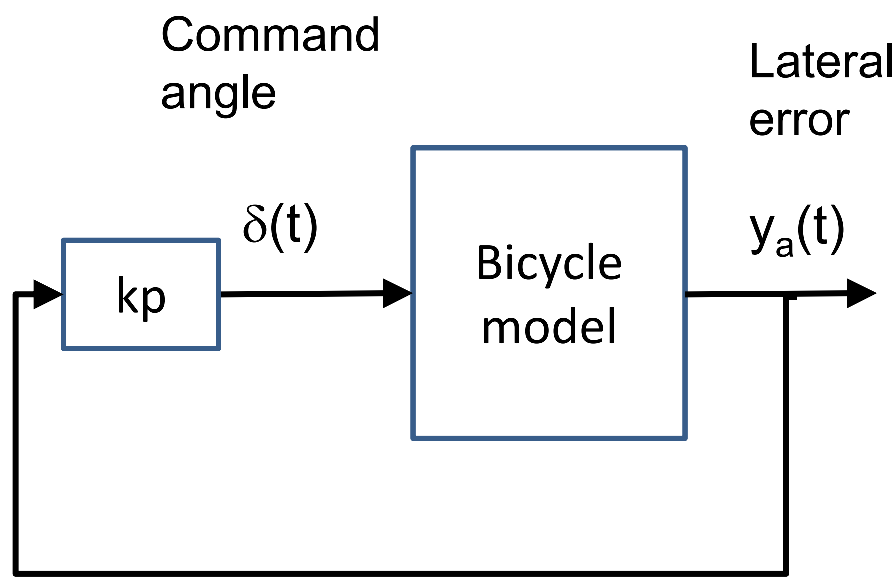


Bicycle Steering Model



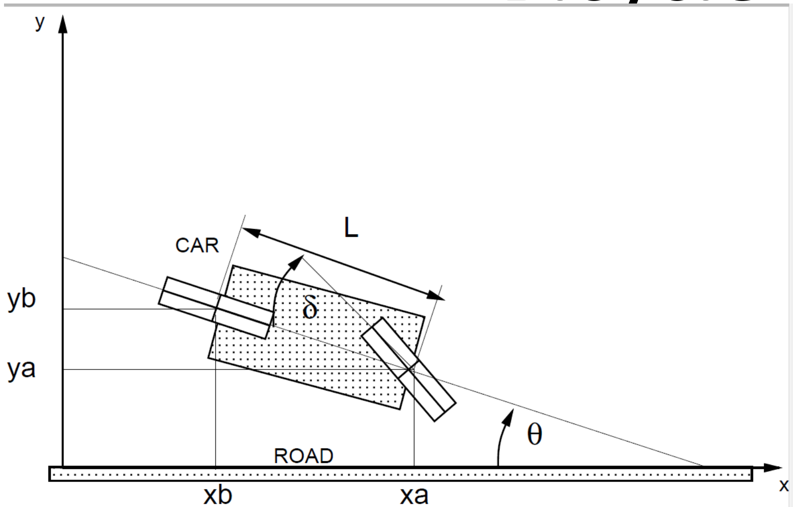
Proportional control:

$$\delta(t) = k_p y_a(t)$$



Check angle in your car, check sign of kp…

Bicycle Steering Model



Proportional control:

$$\delta(t) = k_p y_a(t)$$

$$\ddot{y}_a = \frac{-V^2}{L}\delta(t) - V\dot{\delta}(t).$$

$$\ddot{y}_a + V k_p \dot{y}_a(t) + \frac{V^2}{L} k_p y_a(t) = 0.$$

Laplace transform:

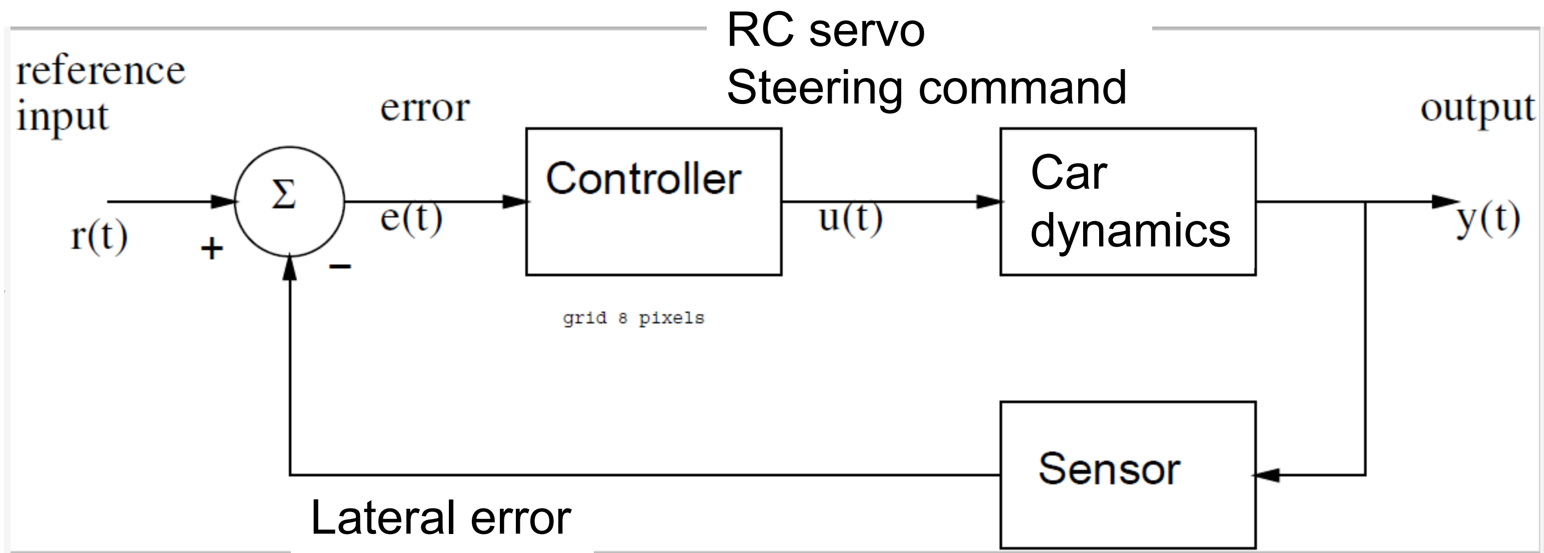
$s^2 Y(s) + V\ k_p\ s\ Y(s) + (V^2/L)\ k_p\ Y(s) =$

$+s\ y(0^-) + y'(0^-)\ + V\ k_p\ y(0^-)$ (initial conditions)

Eigenvalues:

$$\lambda_{1,2} = \frac{V}{2}\left(-k_p \pm \sqrt{k_p^2 - \frac{4k_p}{L}}\right)$$

Steering Control overview



Offset from track
r(t) =0 (mostly)
Where might offset be useful?

Proportional control:
u = kp*e = kp* (r-y);

Proportional + derivative control:

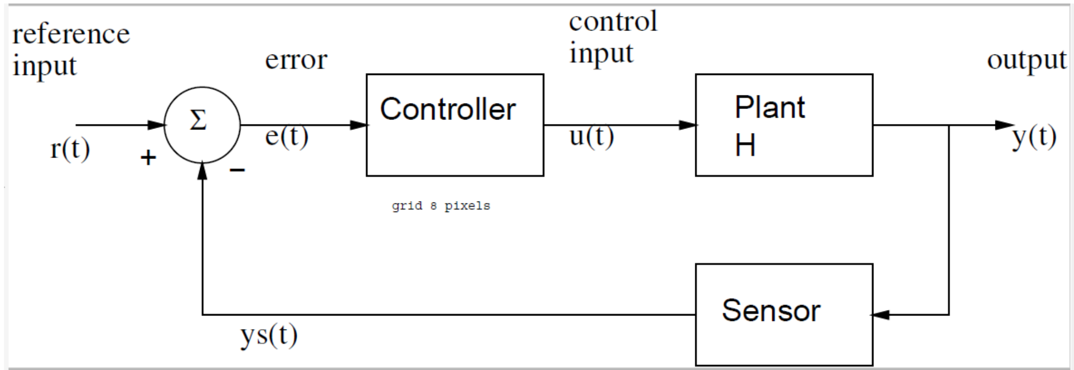
```
u = kp*e + kd * y_dot;
y_dot = (y - y_old)/T;
```

Check sign for kp….

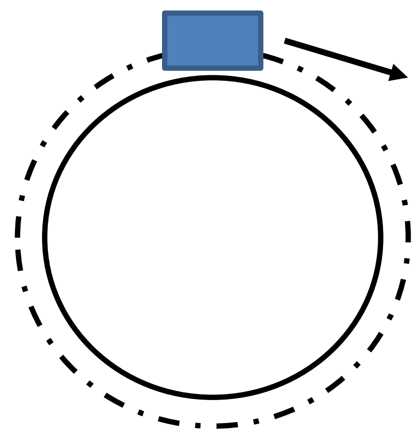
Proportional + integral control

```
u = kp*e + ki * e_sum;
e_sum = e_sum + e;
```

Bicycle Steering Control- proportional control



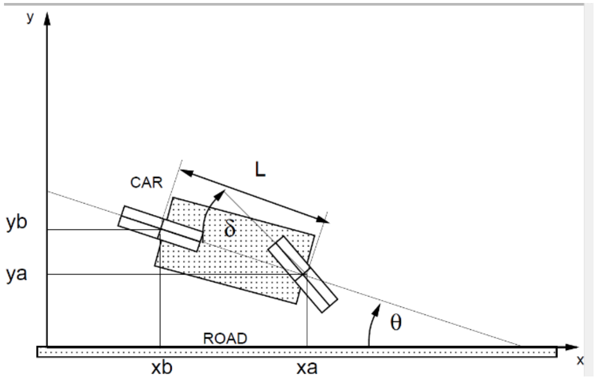
Note steady state error:
car follows larger radius



Proportional control:
r = 0 (to be on straight track)
$\delta = u = k_p * e$

Note: steady state error

Bicycle Steering Model



Proportional control: $\delta(t) = k_p y_a(t)$

$$\ddot{y}_a + V k_p \dot{y}_a(t) + \frac{V^2}{L} k_p y_a(t) = 0.$$

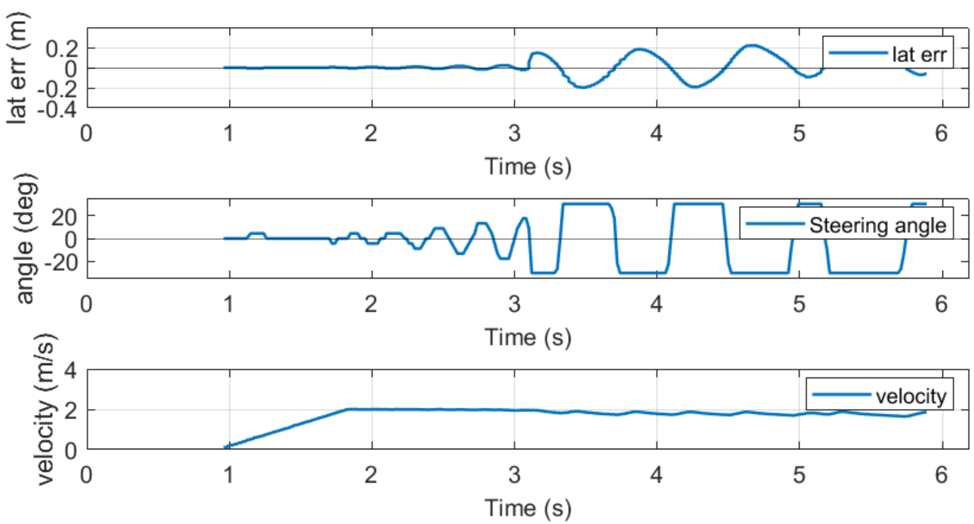
Eigenvalues: $$\lambda_{1,2} = \frac{V}{2}\left(-k_p \pm \sqrt{k_p^2 - \frac{4k_p}{L}}\right)$$

Critical damping: $\lambda_1 = \lambda_2$ ➔

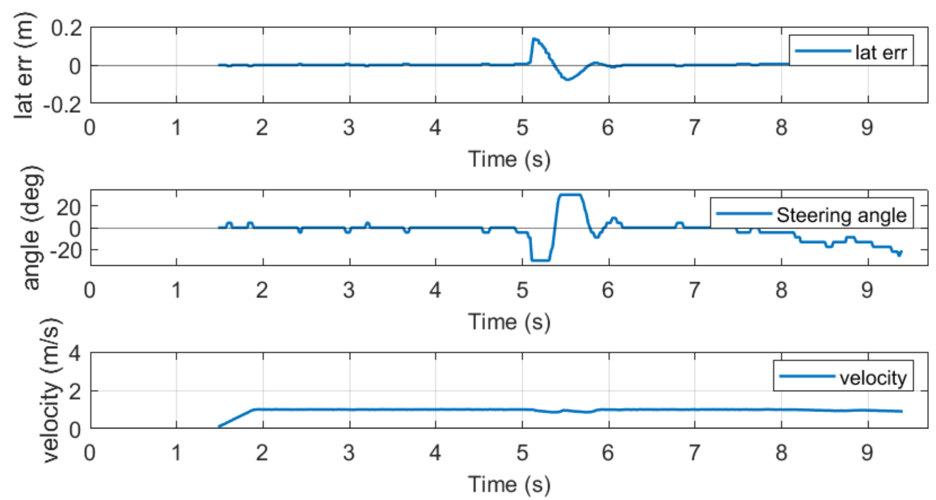
$k_p^2 = 4\ k_p/L$ or $k_p = 4/L = 4/0.3$ m = 13 rad/m = 760 deg/m

At 2 m/s, doesn't work well- servo saturates, also simulation dynamics…

2 m/s k_p =800 deg/m Kd = 0



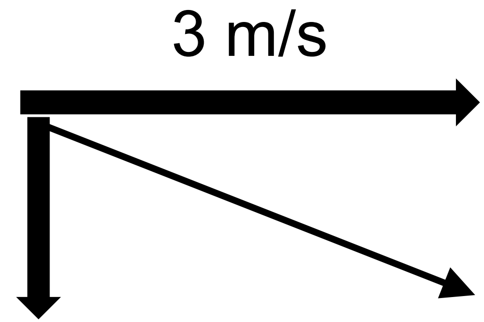
1 m/s k_p =800 deg/m Kd = 0

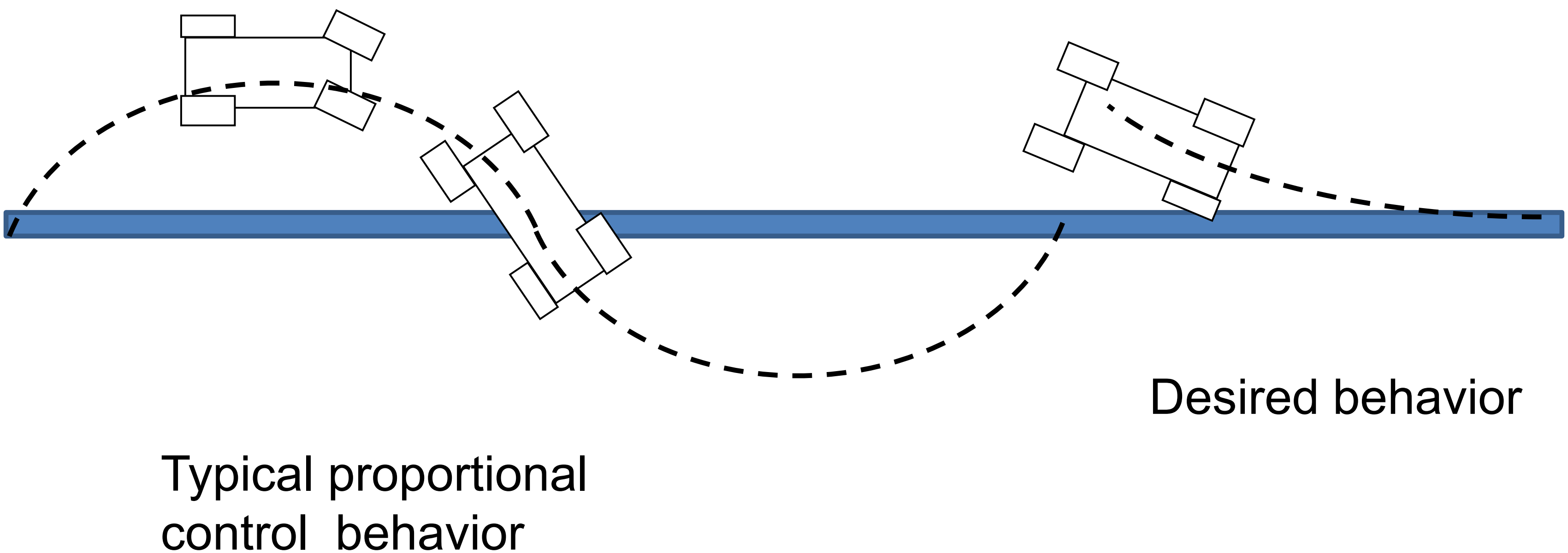


PD control motivation

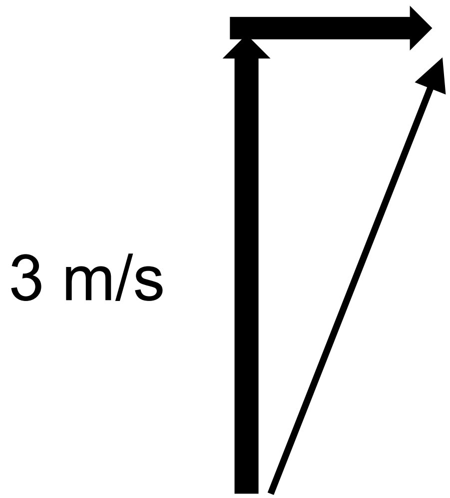
Proportional control $\delta = u = k_p * e$

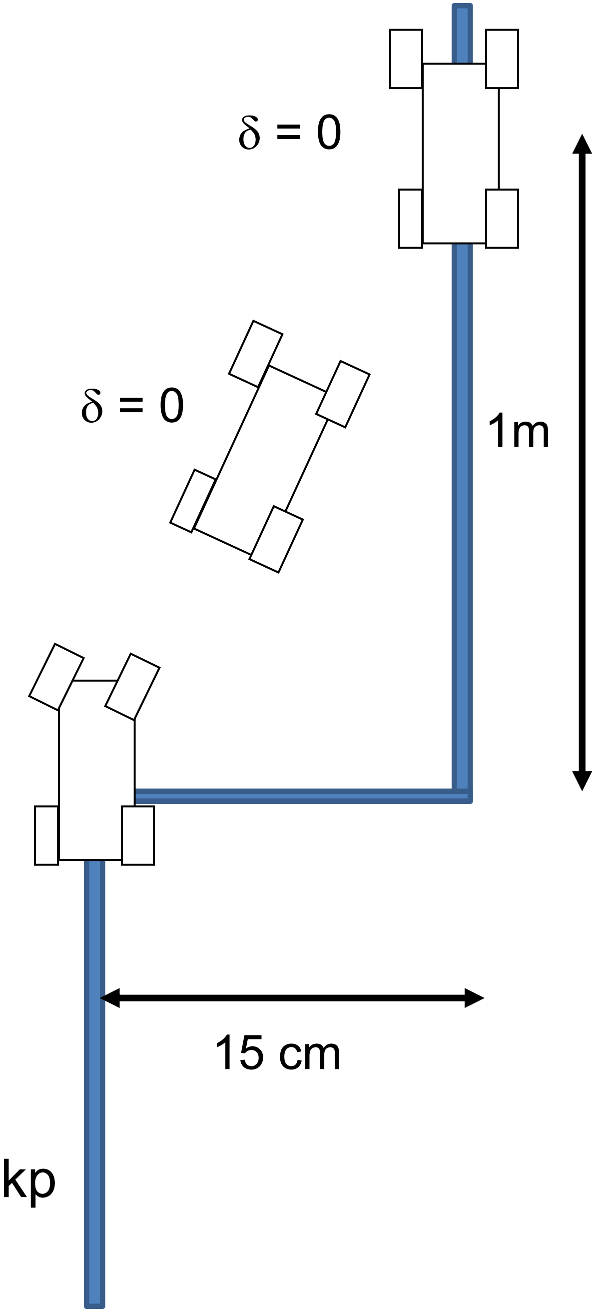
$e = ?$





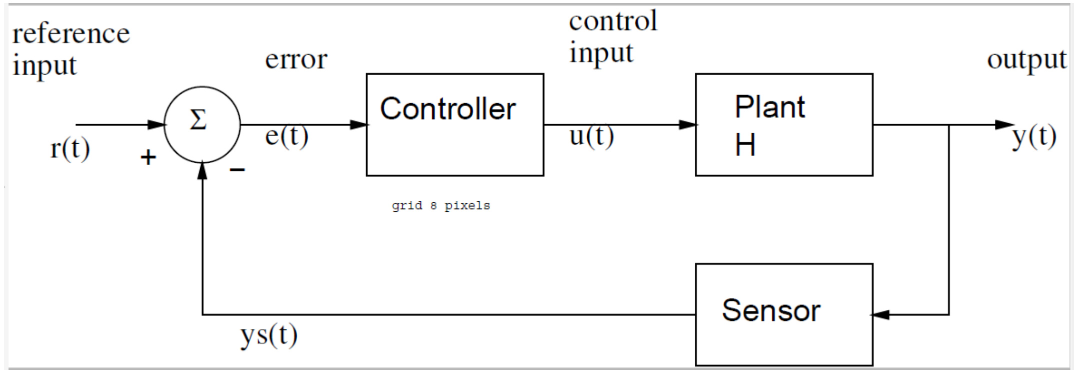
PD parameters



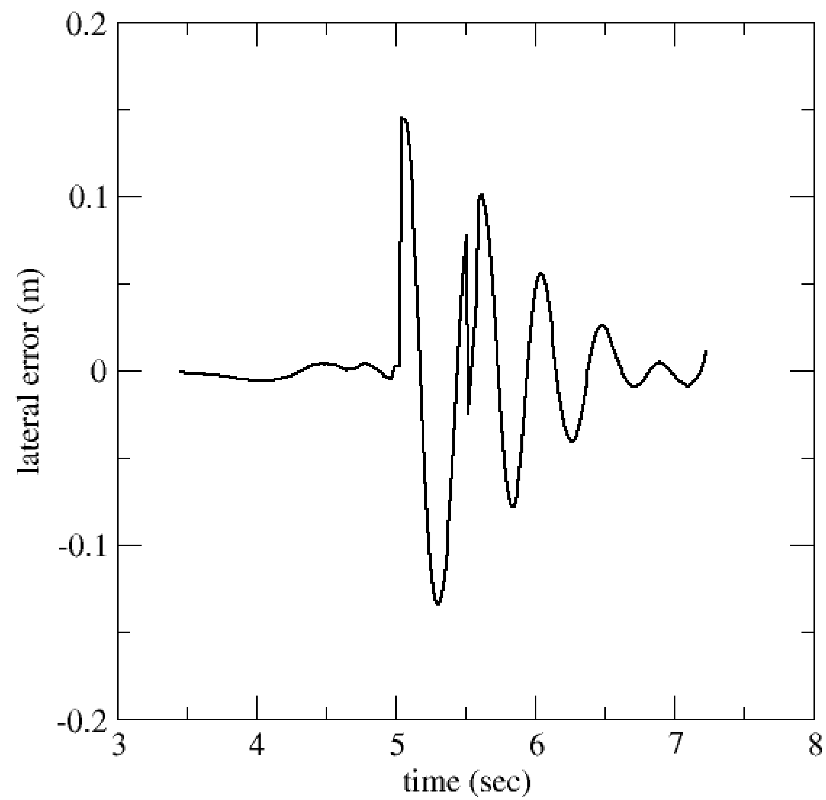


Step: 15 cm. Speed 3 m/s
Choose step response 1m ~ 300 ms
Then lateral velocity = 15 cm/300 ms = 0.5 m/sec
At mid point:
δ = 0 = kp 7.5 cm + kd 0.5 m/sec ➔ kd ~ [0.15 sec] kp

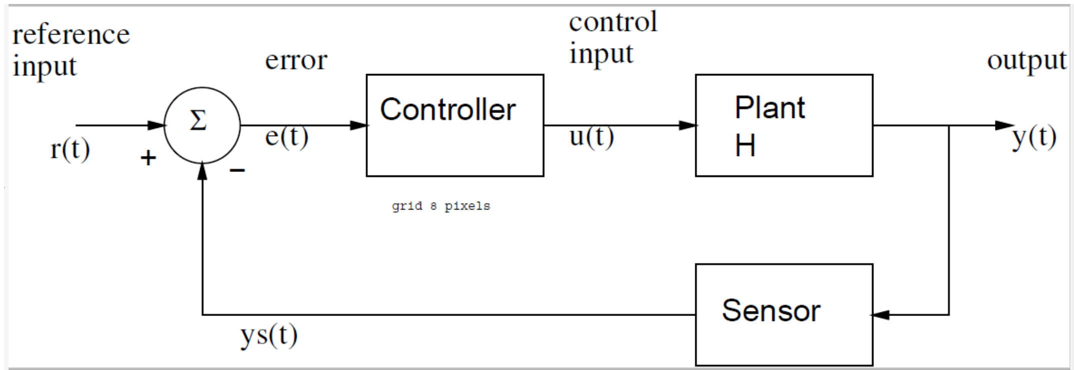
Steering Control- PD

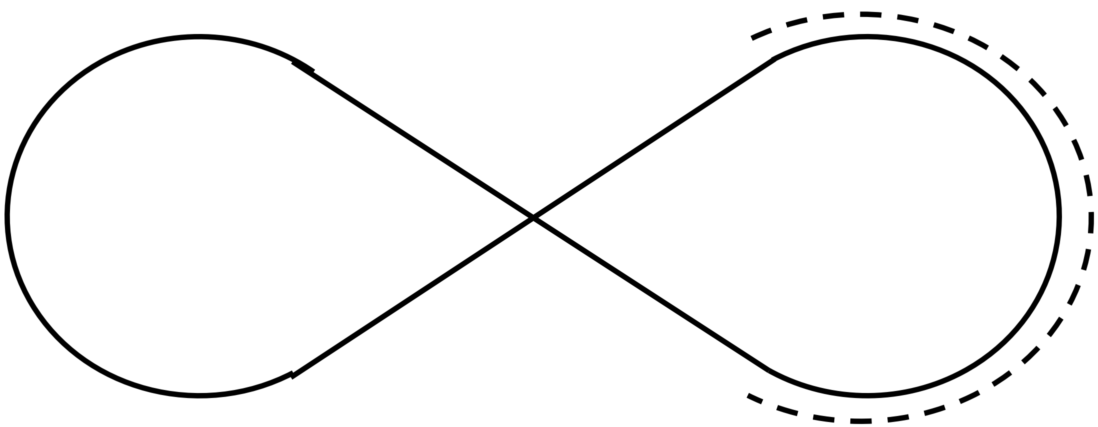


Example under-damped steering:

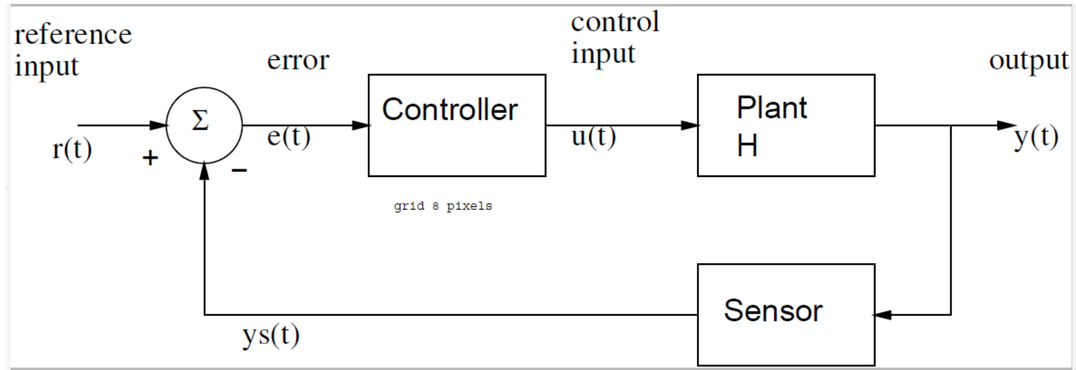


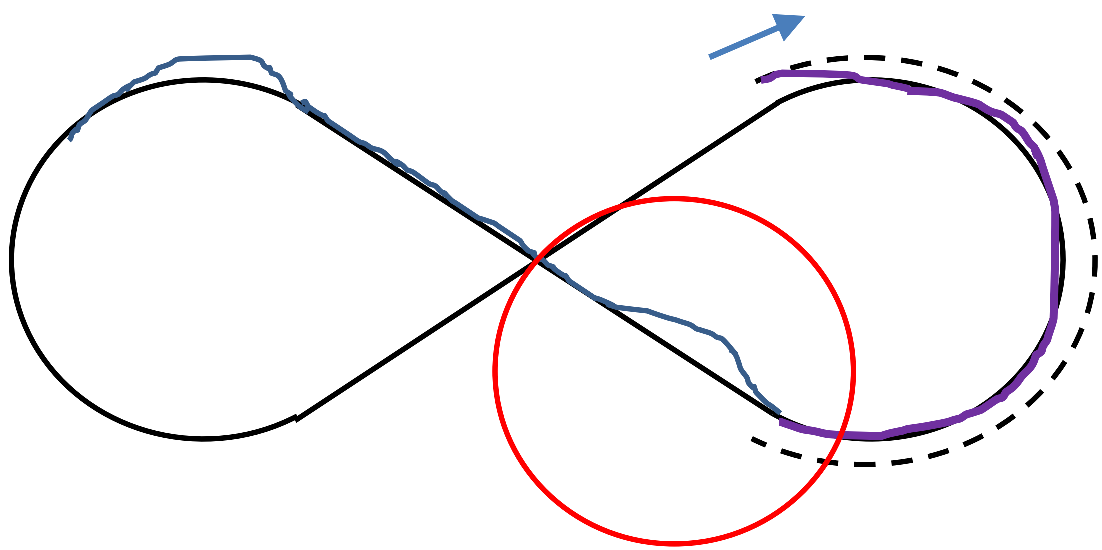
Proportional + Integral





Proportional + Integral

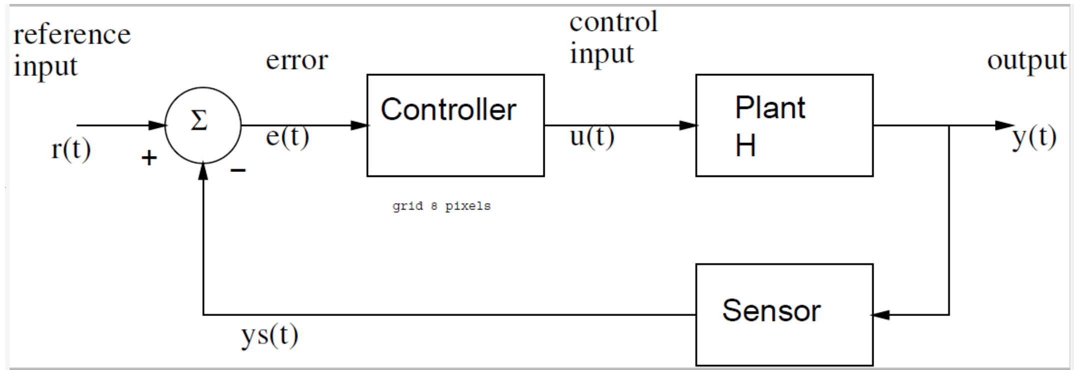


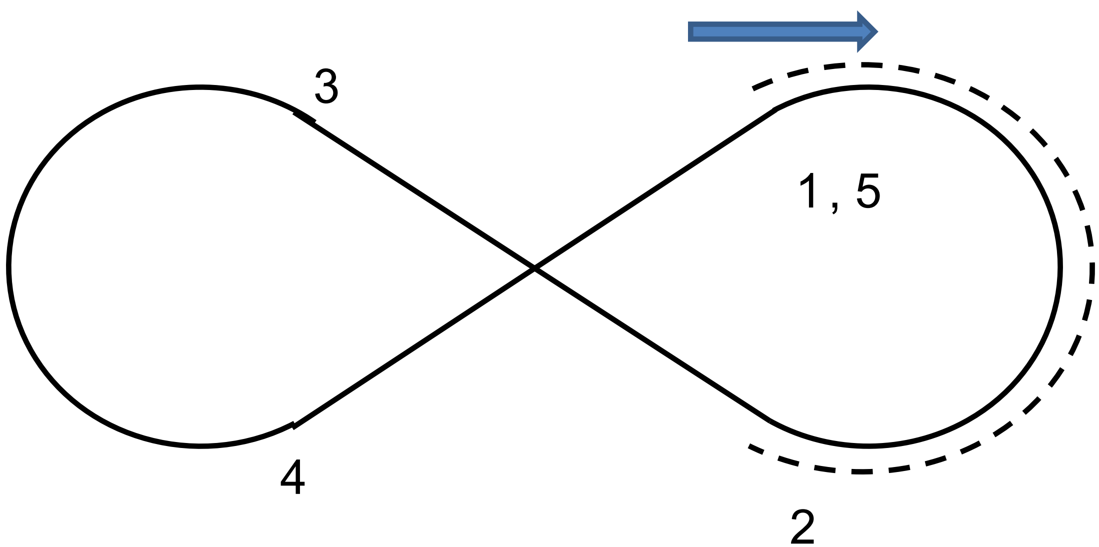


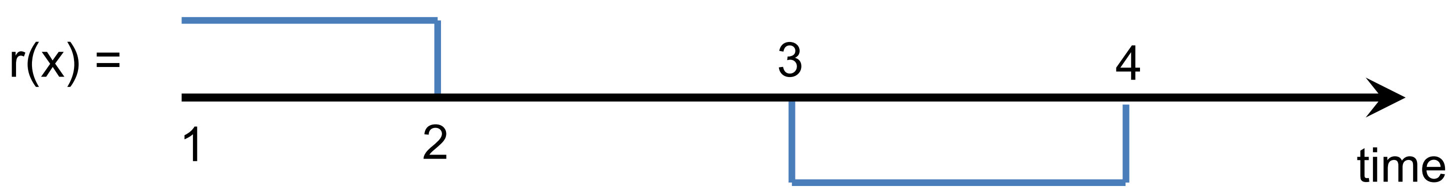
P+I control: delta =kp e + ki (integral e)
P control: delta = kp e

Anti-windup needed here

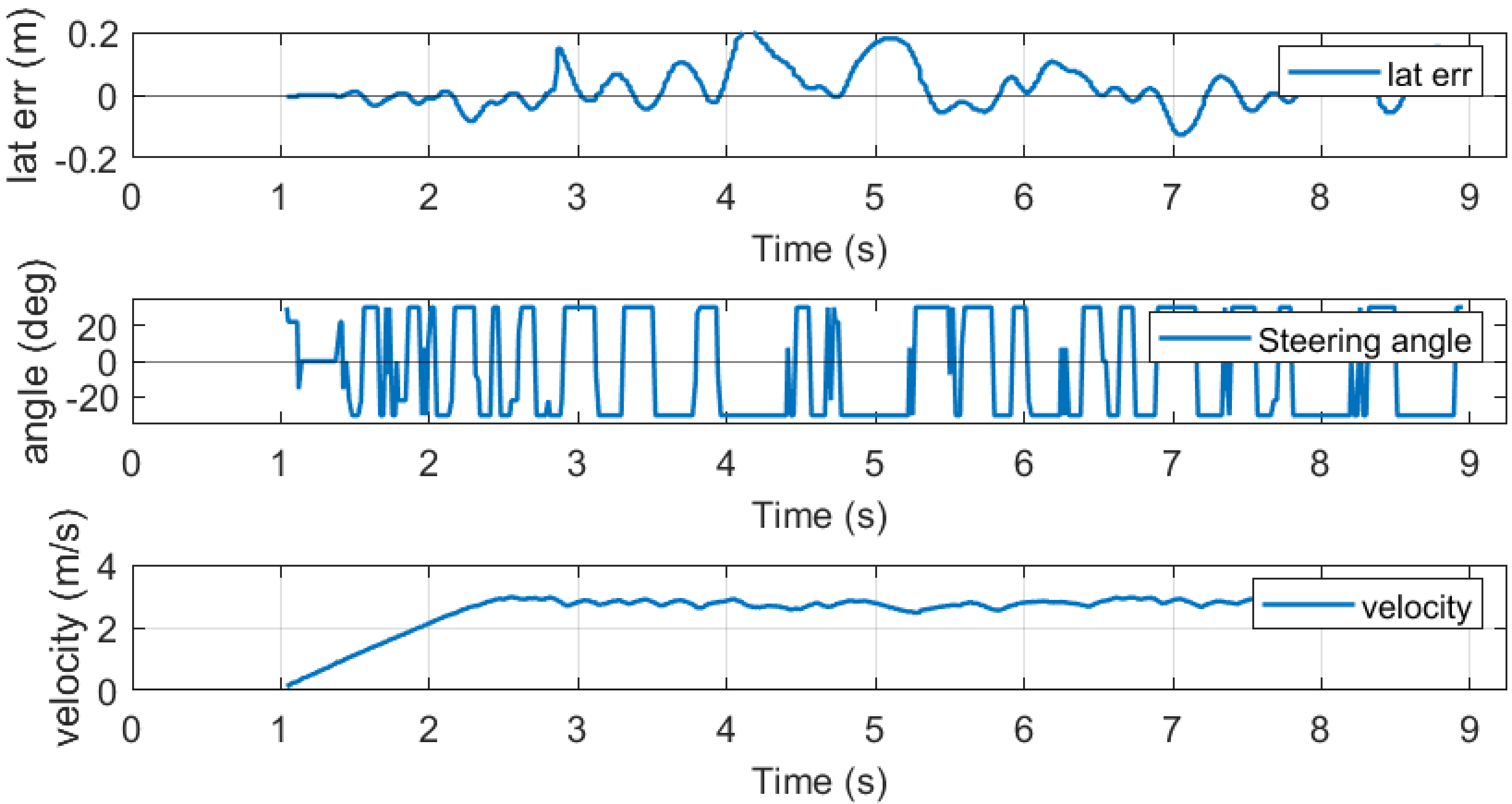
Feedforward



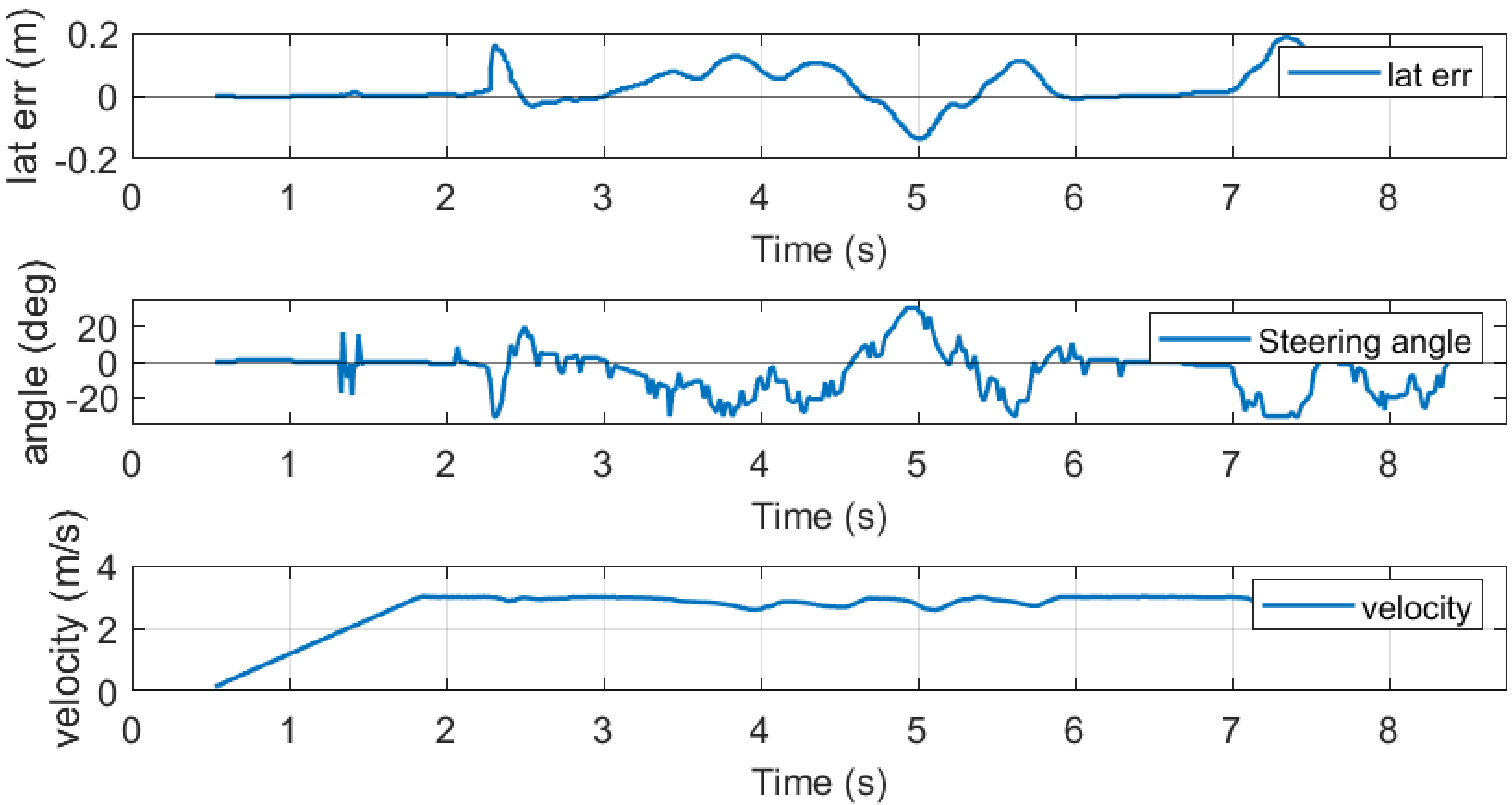




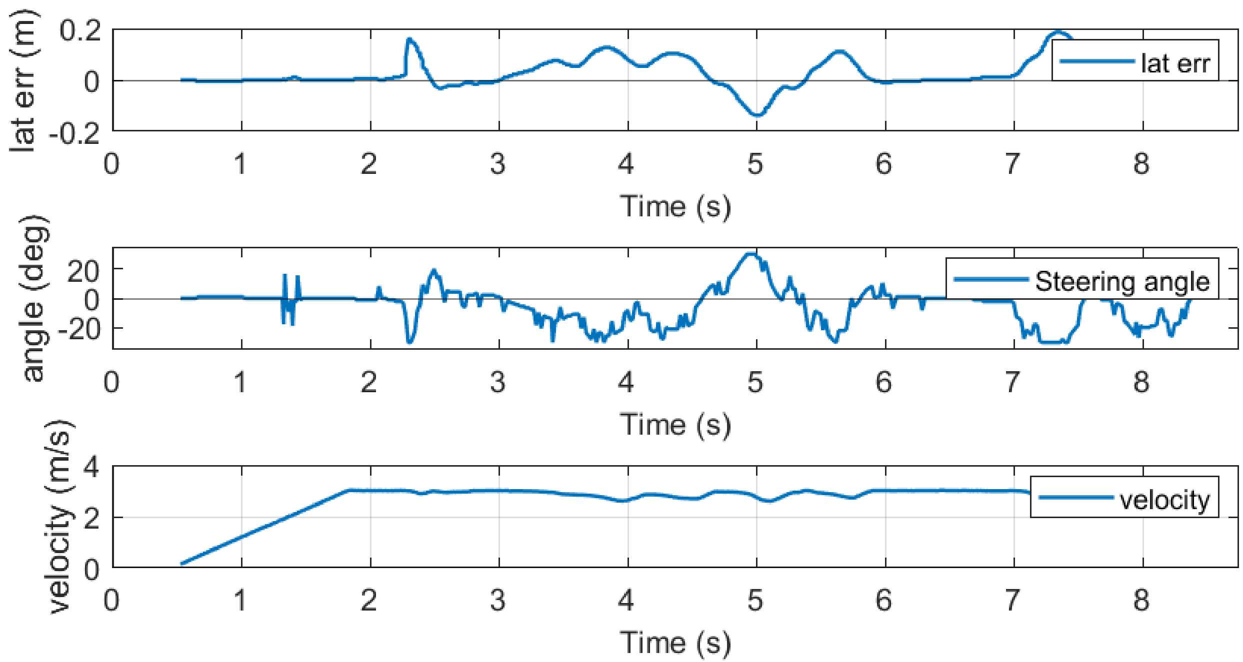
Proportional + derivative control.
Kp = 4000 deg/m, 70 rad/m
Kd = 1000 deg/(m/sec)
V=3 m/s, slew rate 600 deg/0.16 sec
NOTE: = bang-bang!
What is problem with bang bang?
Break servo, nonlinear (unstable)



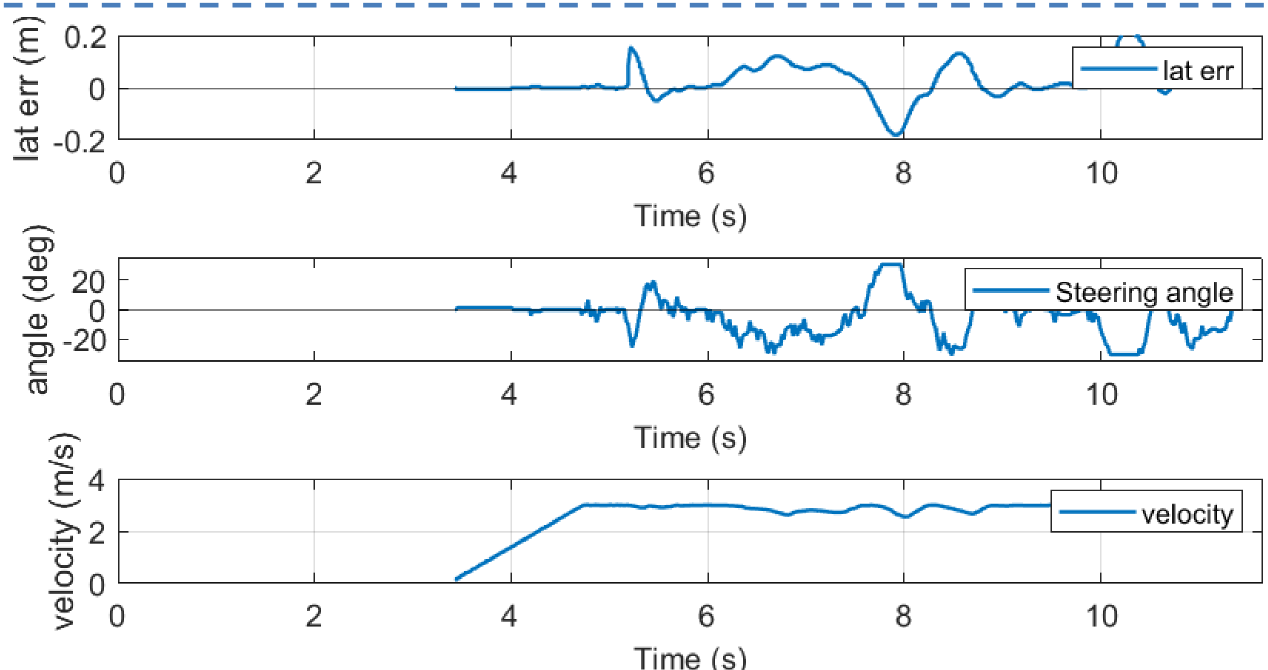
Proportional + derivative control.
Kp = 200 deg/m,
Kd = 30 deg/(m/sec) = (0.15 sec) Kp
V=3 m/s, slew rate 600 deg/0.16 sec
NOTE: = not bang-bang



Kp = 200 deg/m, Kd = 30 deg/(m/sec). V=3 m/s



Slew 600 deg/160 ms



Slew 60 deg/160 ms

EECS192 Lecture 7
Motor Modelling and Steering Introduction
Mar. 2, 2021

Topics

- Checkpoint 5: feedback
- Checkpoint 6: closed-loop control
- Time spent survey
- HW 1 Notes
- PWM for motor drive/MOSFET intro
- Steering control II
- Embedded Issues: deadlock

C.O.P. Watchdog timer

- Despite extensive software and hardware testing, faults will still occur in real devices. Even momentary noise spikes on a power supply can lock up a processor occasionally. Such events will occur on the power grid several times a year. Watchdog timers provide a last line of defense to prevent system failure with minimal hardware cost.
- https://developer.mbed.org/cookbook/WatchDog-Timer

ESP32 Watchdog

The Interrupt Watchdog is responsible for detecting instances where FreeRTOS task switching is blocked for a prolonged period of time. The TWDT is responsible for detecting instances of tasks running without yielding for a prolonged period.

Neither critical sections or interrupt handlers should ever block waiting for another event to occur.

This is a symptom of CPU starvation and is usually caused by a higher priority task looping without yielding to a lower-priority task thus starving the lower priority task from CPU time. This can be an indicator of poorly written code that spinloops on a peripheral, or a task that is stuck in an infinite loop.

The TWDT is built around the Hardware Watchdog Timer in Timer Group 0.

- TIMGn_Tx_INT_WDT_INT: Generated when a watchdog timer interrupt stage times out.

Resource sharing, e.g. printf, timer

Log_add uses queue, so only access to UART is from

```
xQueueReceive(log_queue, log,
                    portMAX_DELAY);
printString(log);
```

Note: if using multiple printf() in different tasks, can contend for UART

Standard practice: Have access to each peripheral only through a single handler function, with queues for communication.

Non-blocking print

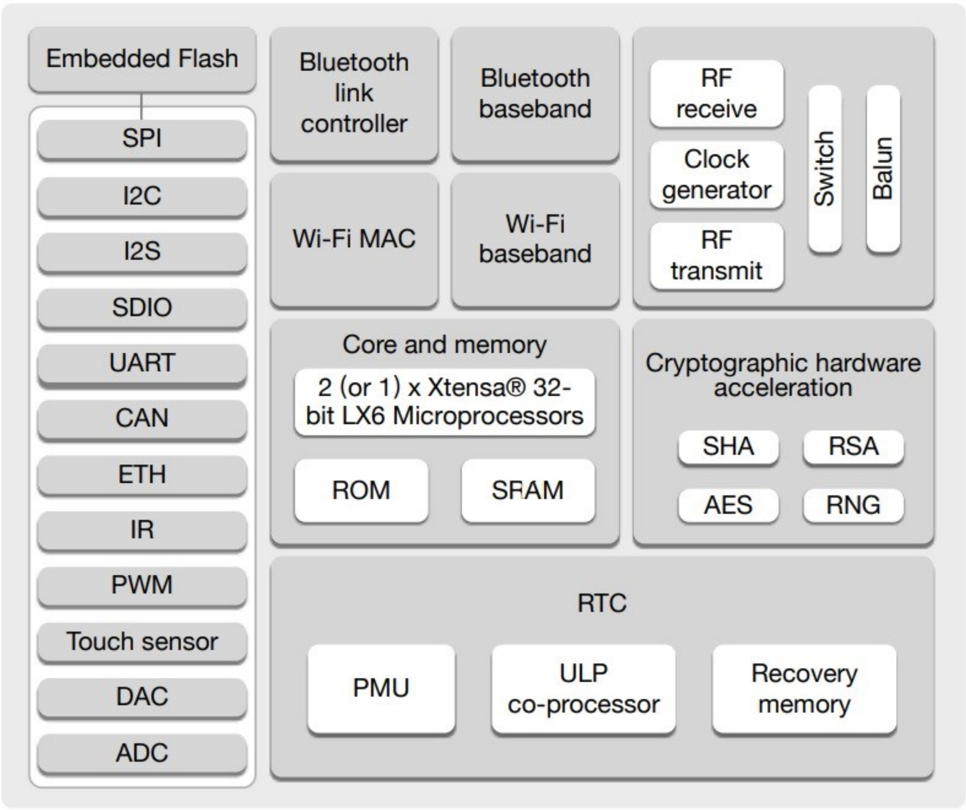
https://github.com/ucb-ee192/SkeletonHuzzah32

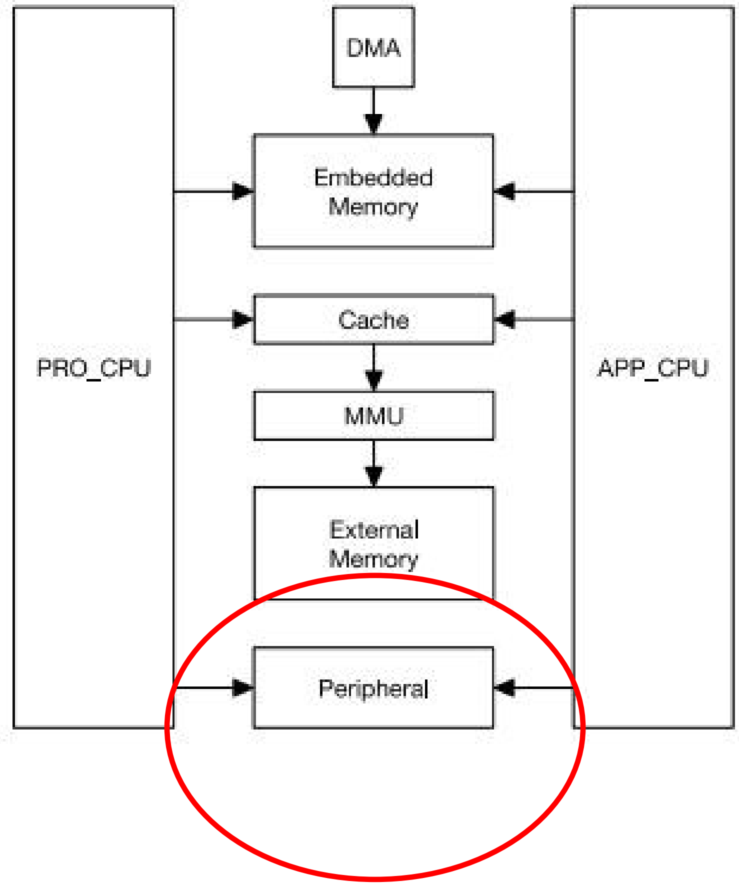
```
snprintf(log, sizeof(log),
          "Idle. sum of cos = %d \n",(long)ZSum);
log_add(log);
```

← if snprintf is too slow, can use itoa(), etc

```
#define WIFILOG  // choose UDP instead of UART
void log_add(char *log)
{ xQueueSend(log_queue, log, 0);
     // send data to back of queue,
    // non-blocking, wait=0 ==> return
immediately if the queue is already full.
}
static void uart_log_task(void *pvParameters)
{  …
xQueueReceive(log_queue, log, portMAX_DELAY);
```

Peripheral Sharing

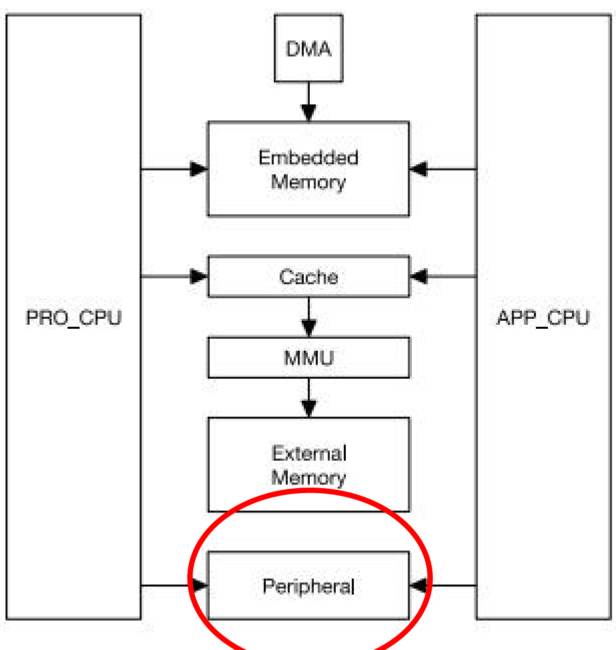




Timer Registers- Sharing

Timer 1 configuration and control registers

TIMGn_T1CONFIG_REG Timer 1 configuration register
TIMGn_T1LO_REG Timer 1 current value, low 32 bits
TIMGn_T1HI_REG Timer 1 current value, high 32 bits
TIMGn_T1UPDATE_REG
Write to copy current timer value to TIMGn_T1_(LO/HI)_REG
TIMGn_T1ALARMLO_REG Timer 1 alarm value,
TIMGn_T1ALARMHI_REG Timer 1 alarm value, high 32 bits
TIMGn_T1LOADLO_REG Timer 1 reload value, low 32 bits
TIMGn_T1LOAD_REG
Write to reload timer from TIMGn_T1_(LOADLOLOADHI)_REG



Timing execution time

```
timer_get_counter_value(TIMER_GROUP_0, TIMER_0,
                        &task_counter_value0);

[ routine to time goes here]

timer_get_counter_value(TIMER_GROUP_0, TIMER_0,
                        &task_counter_value1);


runtime = ((double) (task_counter_value1-task_counter_value0) / TIMER_SCALE);
 // do floating point after timing

snprintf(log, sizeof(log), “tick=%8.3f milliseconds (s)\n",
       1000*(runtime));
```

Timer mutex (mutual exclusion)

esp_err_t timer_spinlock_take(timer_group_t *group_num*)

Take timer spinlock to enter critical protect.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- group_num: Timer group number, 0 for TIMERG0 or 1 for TIMERG1

esp_err_t timer_spinlock_give(timer_group_t *group_num*)

Give timer spinlock to exit critical protect.

Return

- ESP_OK Success
- ESP_ERR_INVALID_ARG Parameter error

Parameters

- group_num: Timer group number, 0 for TIMERG0 or 1 for TIMERG1

Timer interface- sharing

https://github.com/espressif/esp-idf/blob/release/v4.2/components/driver/timer.c

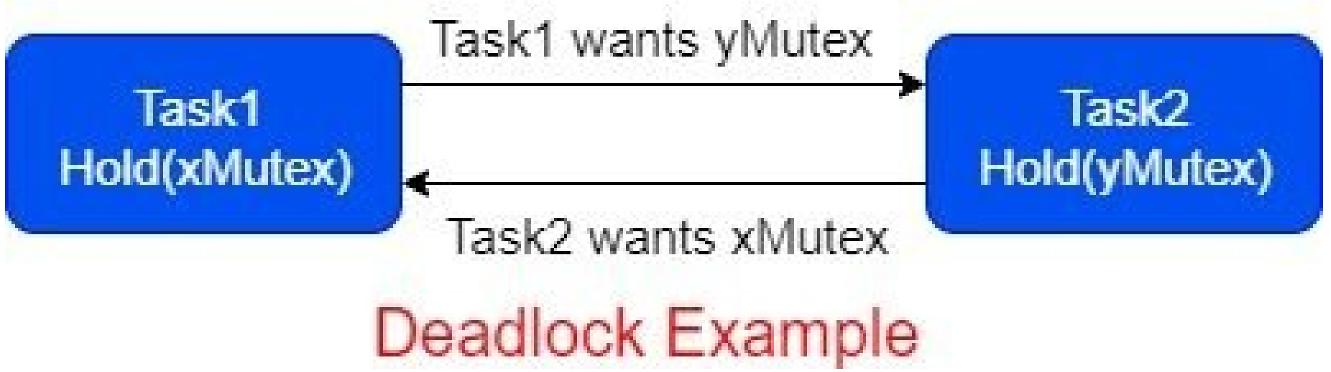
Vanilla FreeRTOS implements critical sections with taskENTER_CRITICAL() which calls portDISABLE_INTERRUPTS()

Note: disabling interrupts is not sufficient – as other core can still interrupt

```
#define TIMER_ENTER_CRITICAL(mux) portENTER_CRITICAL_SAFE(mux);
#define TIMER_EXIT_CRITICAL(mux)  portEXIT_CRITICAL_SAFE(mux);
static portMUX_TYPE timer_spinlock[TIMER_GROUP_MAX] =
    {portMUX_INITIALIZER_UNLOCKED, portMUX_INITIALIZER_UNLOCKED};
```

```
esp_err_t timer_get_counter_value(timer_group_t group_num,
             timer_idx_t timer_num, uint64_t *timer_val)
{ …
TIMER_ENTER_CRITICAL(&timer_spinlock[group_num]);
 timer_hal_get_counter_value(
             &(p_timer_obj[group_num][timer_num]->hal),
             timer_val);
TIMER_EXIT_CRITICAL(&timer_spinlock[group_num]);
 return ESP_OK;
}
```

Deadlock

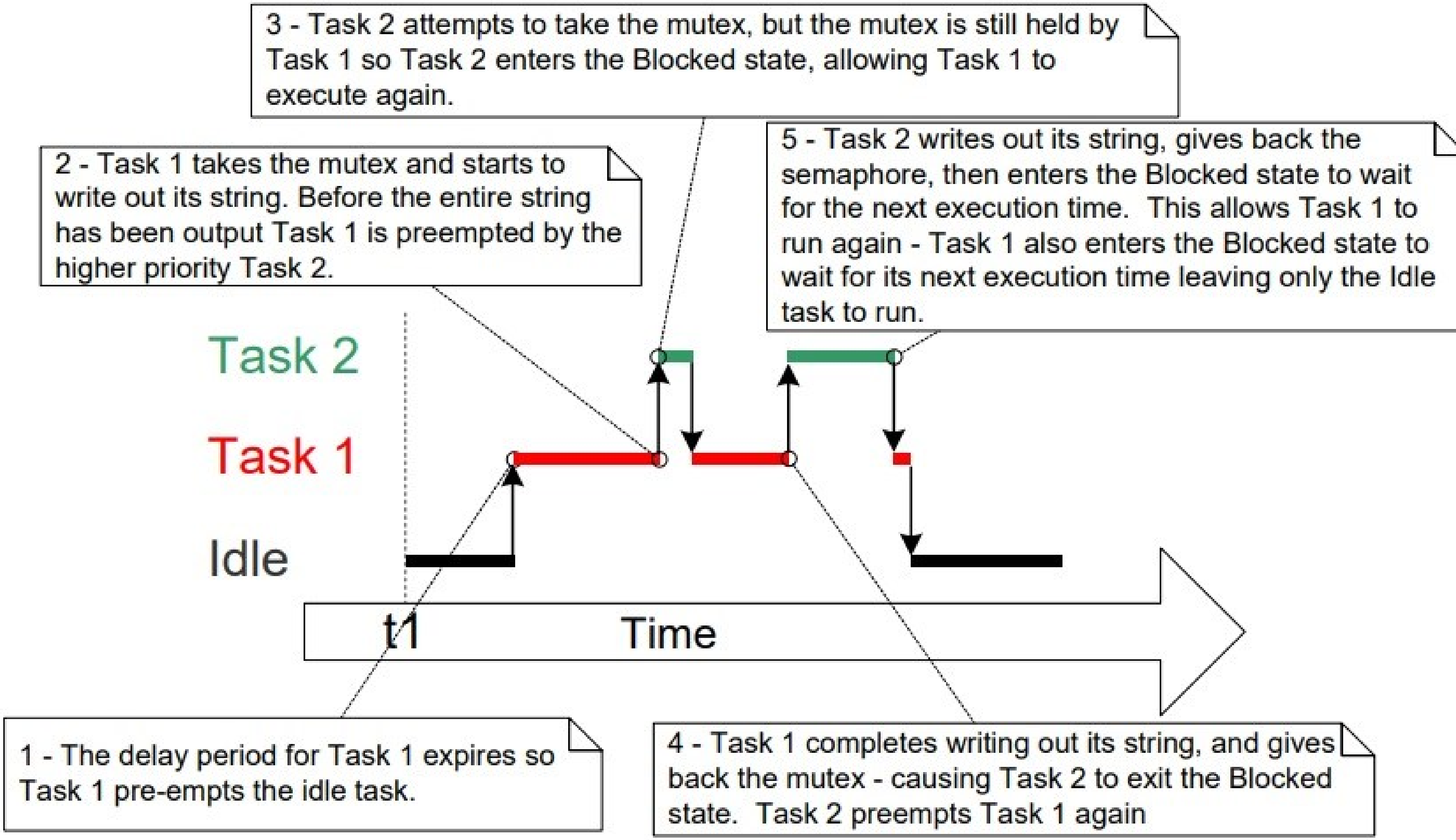


Possible Task1 and Task2 both block
(May be safer to use interrupt if only a single processor…)

Example: xSemaphoreCreateBinary()
See https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
api-reference/system/freertos.html?highlight=priority%20inheritance

Deadlock- print example

(from Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.)



Deadlock

(from Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.

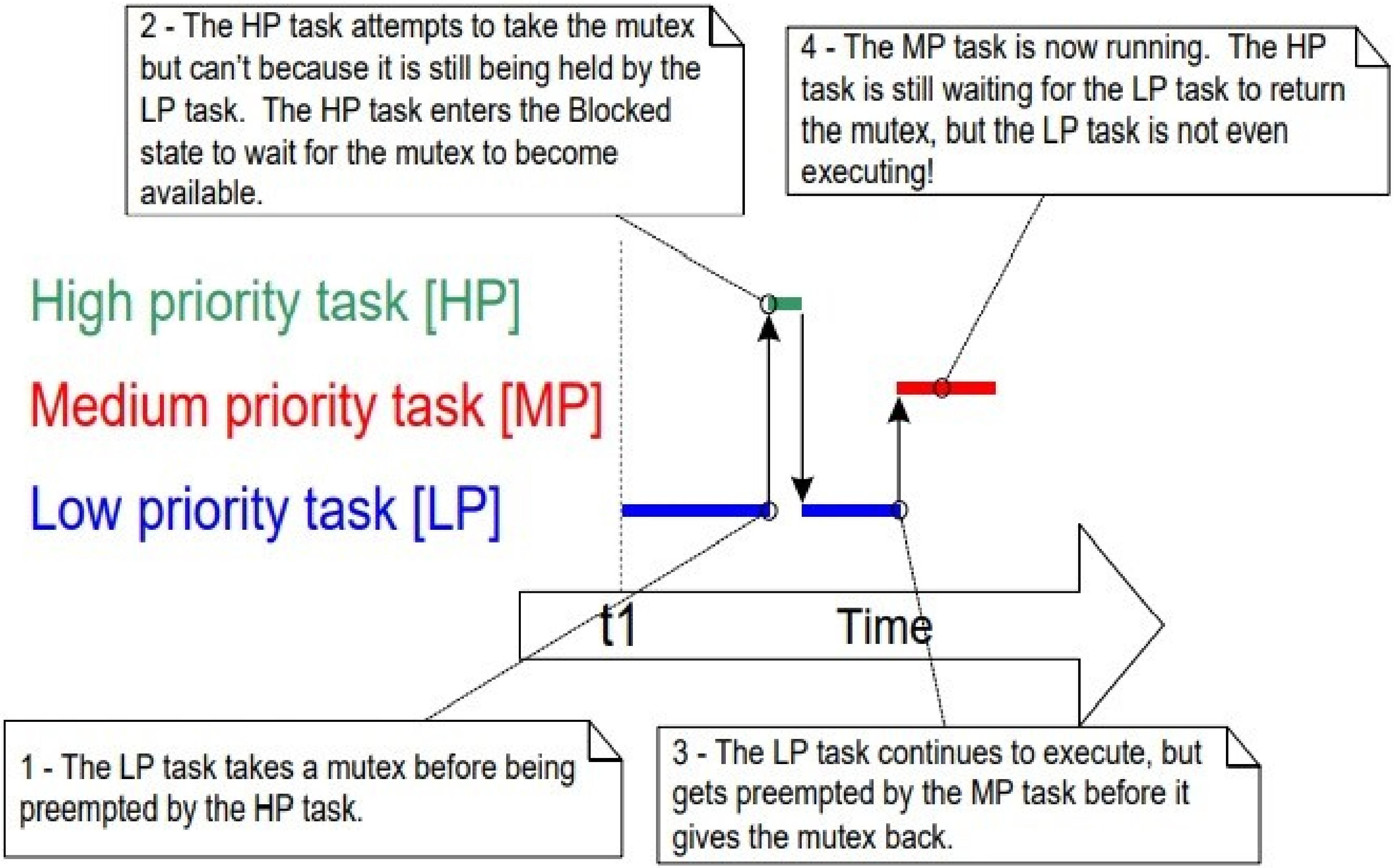


Figure 66. A worst case priority inversion scenario

Deadlock

(from Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.

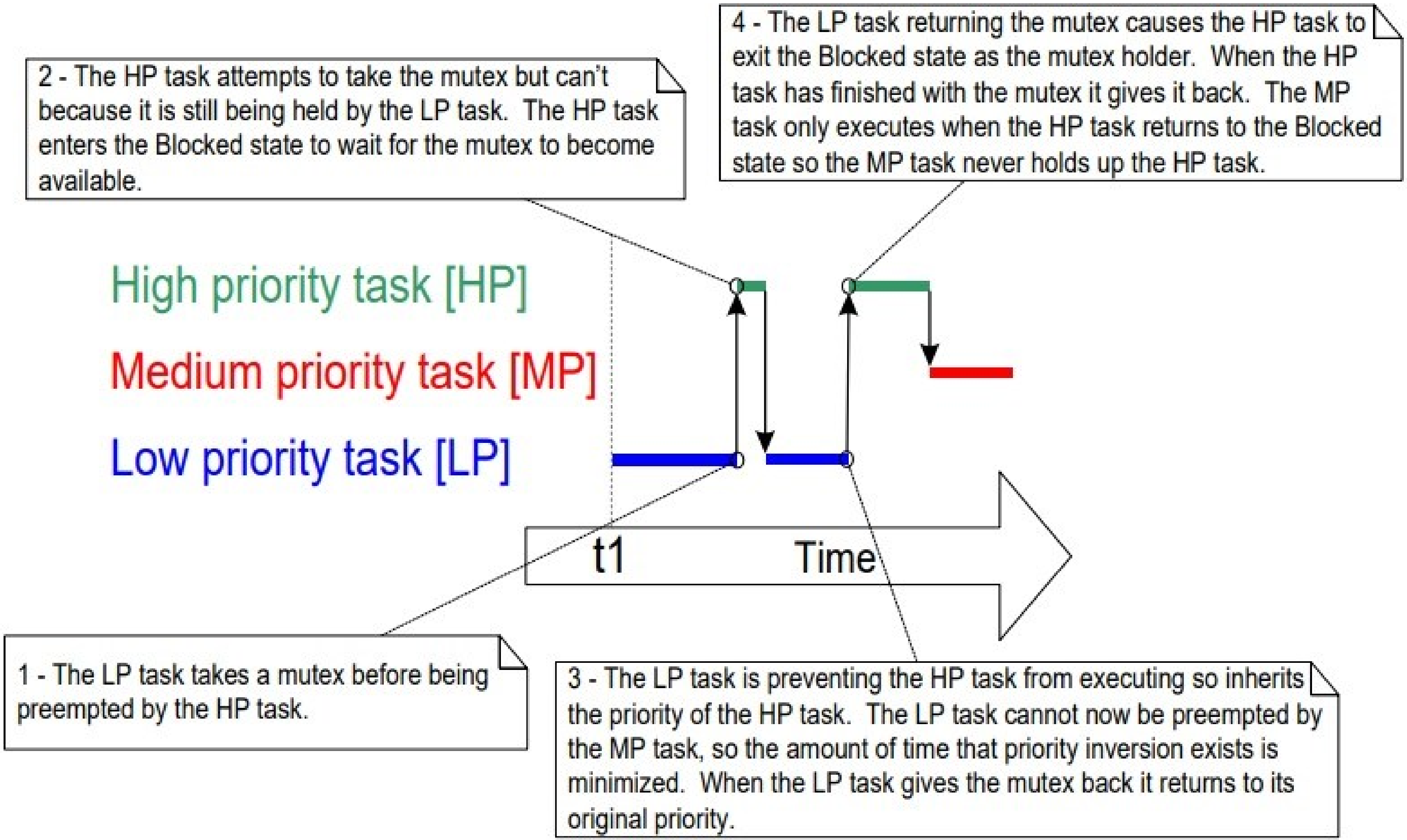


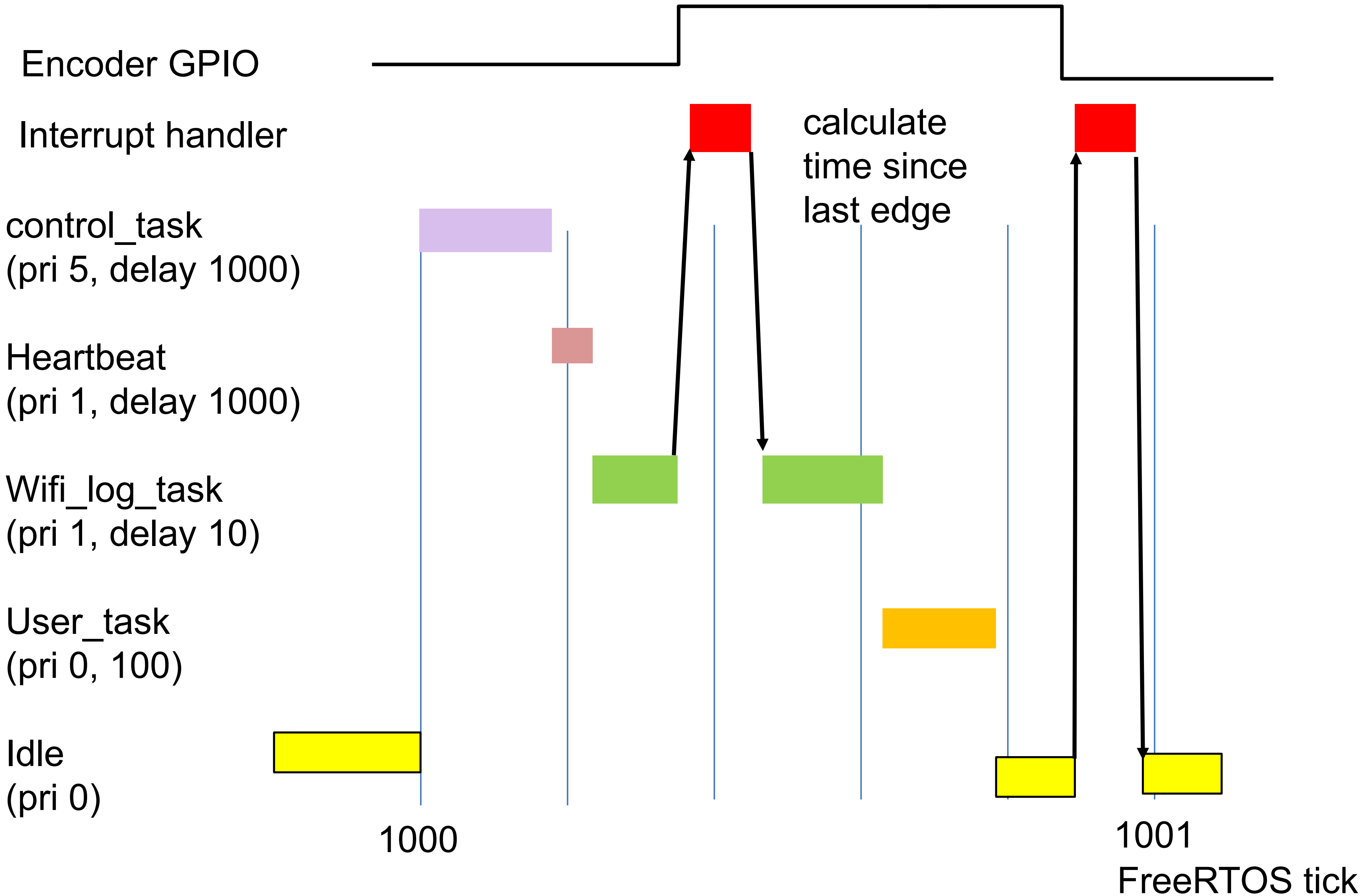
Figure 67. Priority inheritance minimizing the effect of priority inversion

EECS192 Lecture 7
Motor Modelling and Steering Introduction
Mar. 2, 2021

Topics
- Checkpoint 5: feedback
- Checkpoint 6: closed-loop control
- Time spent survey
- HW 1 Notes
- PWM for motor drive/MOSFET intro
- Steering control II
- Embedded Issues: deadlock

Extra Slides

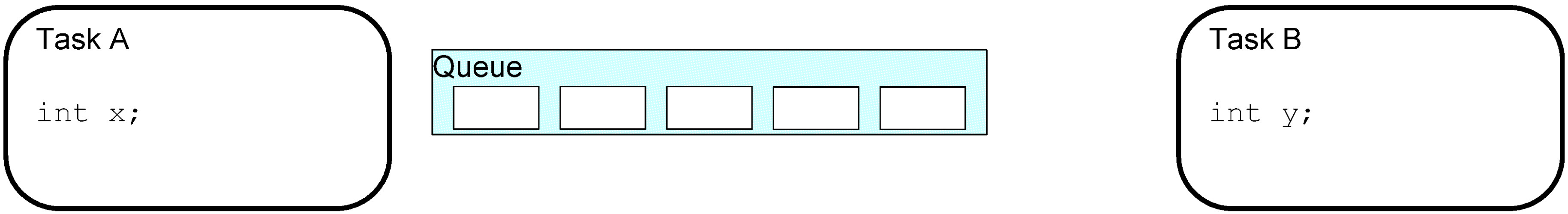
Skeleton Tasks with interrupt



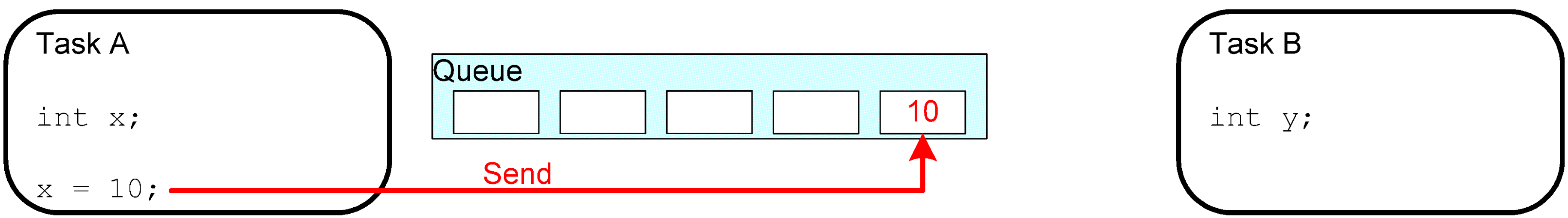
vTaskDelay(delay / portTICK_PERIOD_MS);

Queue in FreeRTOS

161204 Pre-release for FreeRTOS V8.x.x. See http://www.FreeRTOS.org/FreeRTOS-V9.html for information about FreeRTOS V9.x.x. Use http://www.FreeRTOS.org/contact to provide feedback, corrections, and check for updates.



A queue is created to allow Task A and Task B to communicate. The queue can hold a maximum of 5 integers. When the queue is created it does not contain any values so is empty.



Task A writes (sends) the value of a local variable to the back of the queue. As the queue was previously empty the value written is now the only item in the queue, and is therefore both the value at the back of the queue and the value at the front of the queue.

Queue in FreeRTOS

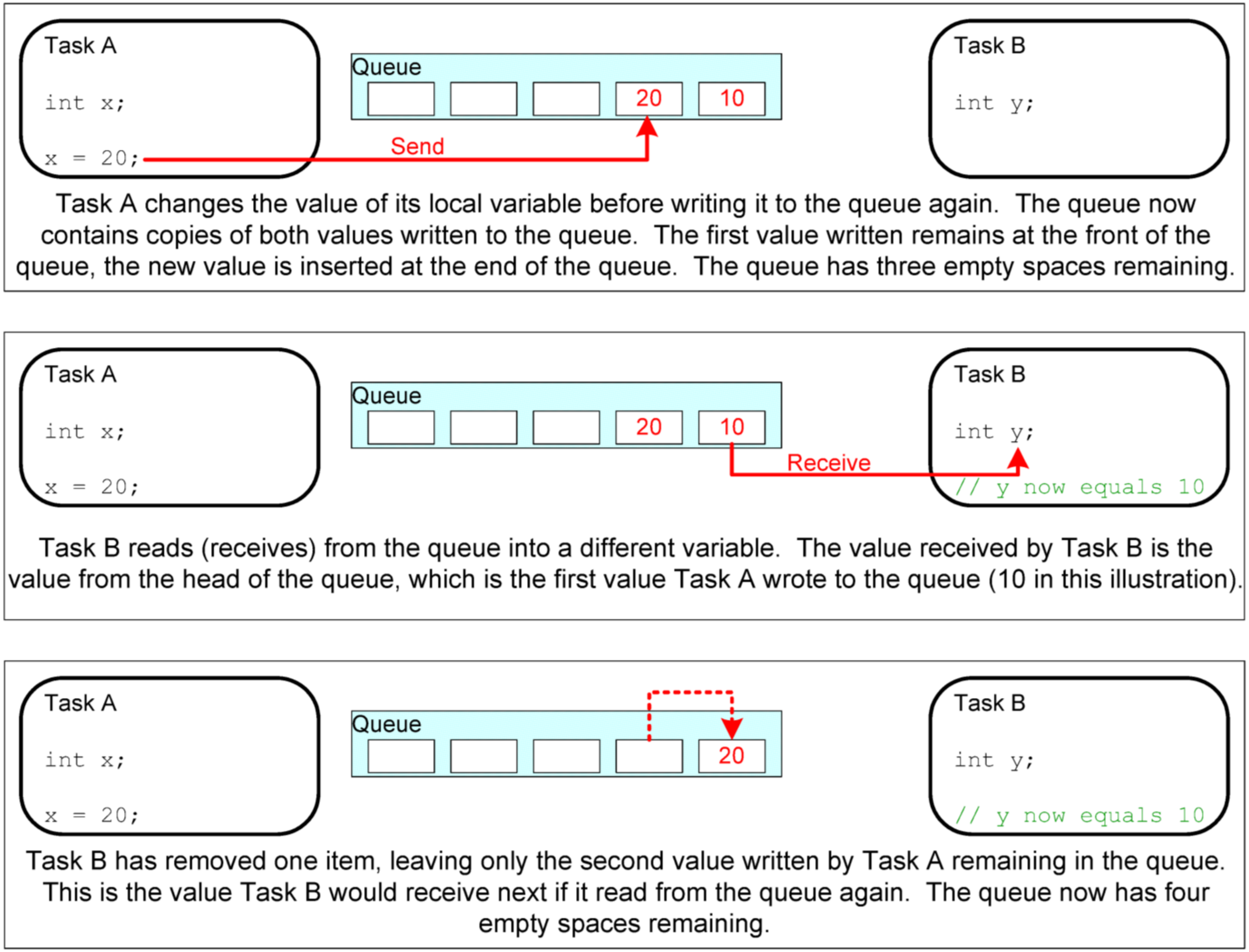
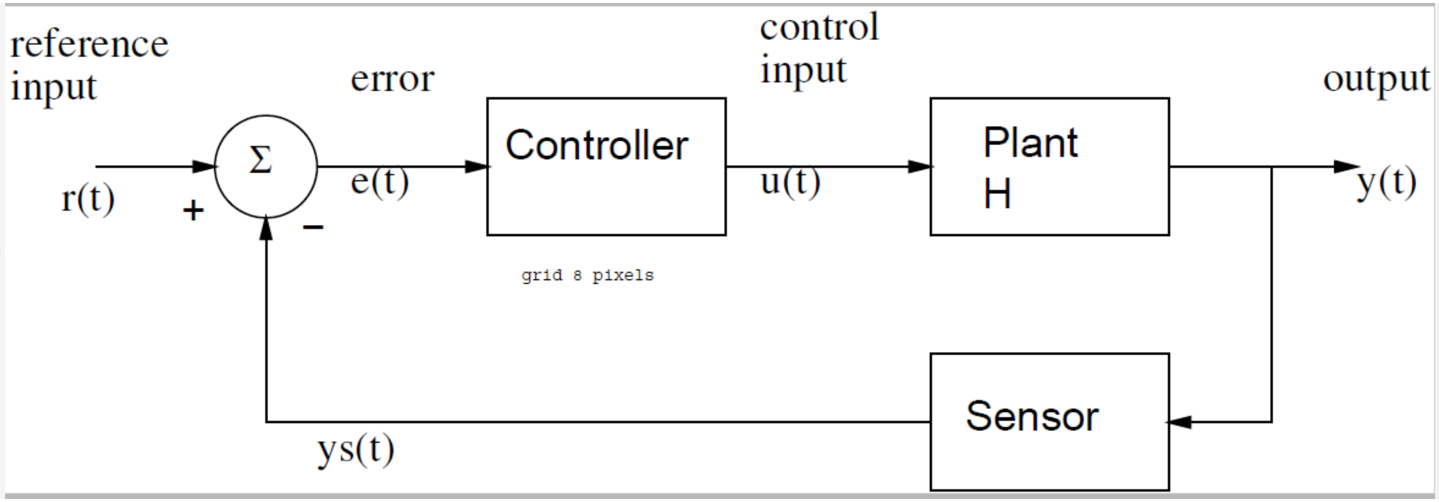


Figure 31. An example sequence of writes to, and reads from a queue

Control Synopsis



State equations: $$\dot{x}(t) = ax(t) + bu(t)$$

Output equations: $$y(t) = cx(t) + du(t)$$

Control Law (P): $$u(t) = k_p e(t) = k_p(r(t) \; - \; y(t)).$$

Control Synopsis

Control Law (P): $u(t) = k_p e(t) = k_p(r(t) \; - \; y(t)).$

New state equations:

$$\dot{x} = ax + bk_p e(t) = ax + bk_p(r - x) = (a - bk_p)x + bk_p r.$$

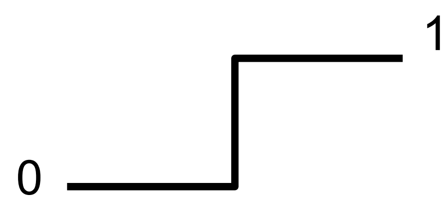
Zero Input Response (non-zero init condx, r(t)=0):

$$x(t) = x(0)e^{(a-bk_p)t} \quad for \quad t \geq 0.$$

$a' = a - b\,k_p$ $\qquad$ $b' = b\,k_p$

Total Response (non-zero init condx) by convolution:

$$x(t_o) = e^{a' t_o} x(0) + \int_0^{t_o} e^{a'(t_o - \tau)} b' r(\tau) d\tau \; . \qquad (10)$$



Step Response (zero init condx) by convolution:

$$x(t_o) = b' \int_0^{t_o} e^{a' t_o} e^{-a'\tau} d\tau \; = \; \frac{-b' e^{a' t_o}}{a'} e^{-a'\tau}|_0^{t_o} \; = \frac{b'}{a'}(1 - e^{-a' t_o}) \; . \qquad (11)$$

Control Synopsis

Control Law (P): $u(t) = k_p e(t) = k_p(r(t) \ - \ y(t)).$

New state equations:

$$\dot{x} = ax + bk_p e(t) = ax + bk_p(r - x) = (a - bk_p)x + bk_p r.$$

Zero Input Response (non-zero init condx):

$$x(t) = x(0)e^{(a-bk_p)t} \quad for \quad t \geq 0.$$

a'=a – b k_p b' = b k_p

Total Response (non-zero init condx) by convolution:

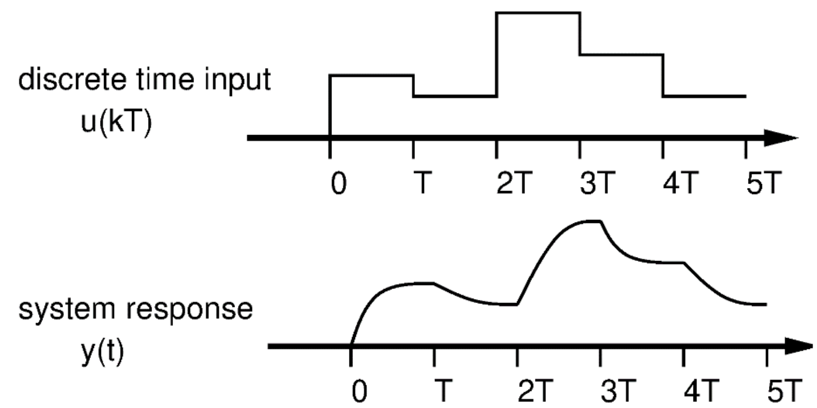
$$x(t_o) = e^{a' t_o} x(0) + \int_0^{t_o} e^{a'(t_o - \tau)} b' r(\tau) d\tau \ . \qquad (10)$$

Step Response (zero init condx) by convolution:

$$x(t_o) = b' \int_0^{t_o} e^{a' t_o} e^{-a'\tau} d\tau \ = \ \frac{-b' e^{a' t_o}}{a'} e^{-a'\tau}|_0^{t_o} \ = \frac{b'}{a'}(1 - e^{-a' t_o}) \ . \qquad (11)$$

Control Synopsis- Discrete Time

Superposition of Step Responses



$$x((k+1)T) = e^{a(k+1)T}x(0) + e^{a(k+1)T}\int_0^{(k+1)T} e^{-a\tau}bu(\tau)d\tau \ . \qquad (15)$$

$$x(kT) = e^{akT}x(0) + e^{akT}\int_0^{kT} e^{-a\tau}bu(\tau)d\tau \ . \qquad (14)$$

$$x((k+1)T) = e^{aT}x(kT) + e^{a(k+1)T}\int_{kT}^{(k+1)T} e^{-a\tau}bu(\tau)d\tau \ = e^{aT}x(kT) + \int_0^T e^{a\lambda}bu(kT)d\lambda \ , \ (16)$$

Control Synopsis- Discrete Time

$$G(T) \equiv e^{aT} \quad \text{and} \quad H(T) \equiv b \int_0^T e^{a\lambda} d\lambda \ . \tag{17}$$

State equations:

$$x((k+1)T) = G(T)x(kT) + H(T)u(kT) \tag{18}$$

Output equations:

$$y(kT) = Cx(kT) + Du(kT) \ . \tag{19}$$

Total Response (non-zero init condx) by convolution:

$$x(k) = G^k x(0) + \sum_{j=0}^{k-1} G^{k-j-1} H u(j) \ . \tag{23}$$

Control Synopsis- Discrete Time

Control Law (P): $U(kT) = k_p\,[r(kT) - x(kT)]$

New state equations:

$$x((k+1)T) = G(T)x(kT) + H(T)k_p(r(kT) - x(kT)) = [G - Hk_p]x(kT) + Hk_p r(kT)\ . \quad (24)$$

$$x((k+1)T) = [e^{aT} + \frac{k_p}{a}(1 - e^{aT})]x(kT) + Hk_p r(kT) \ = \ G'x(kT) + Hk_p r(kT)\ . \quad (25)$$

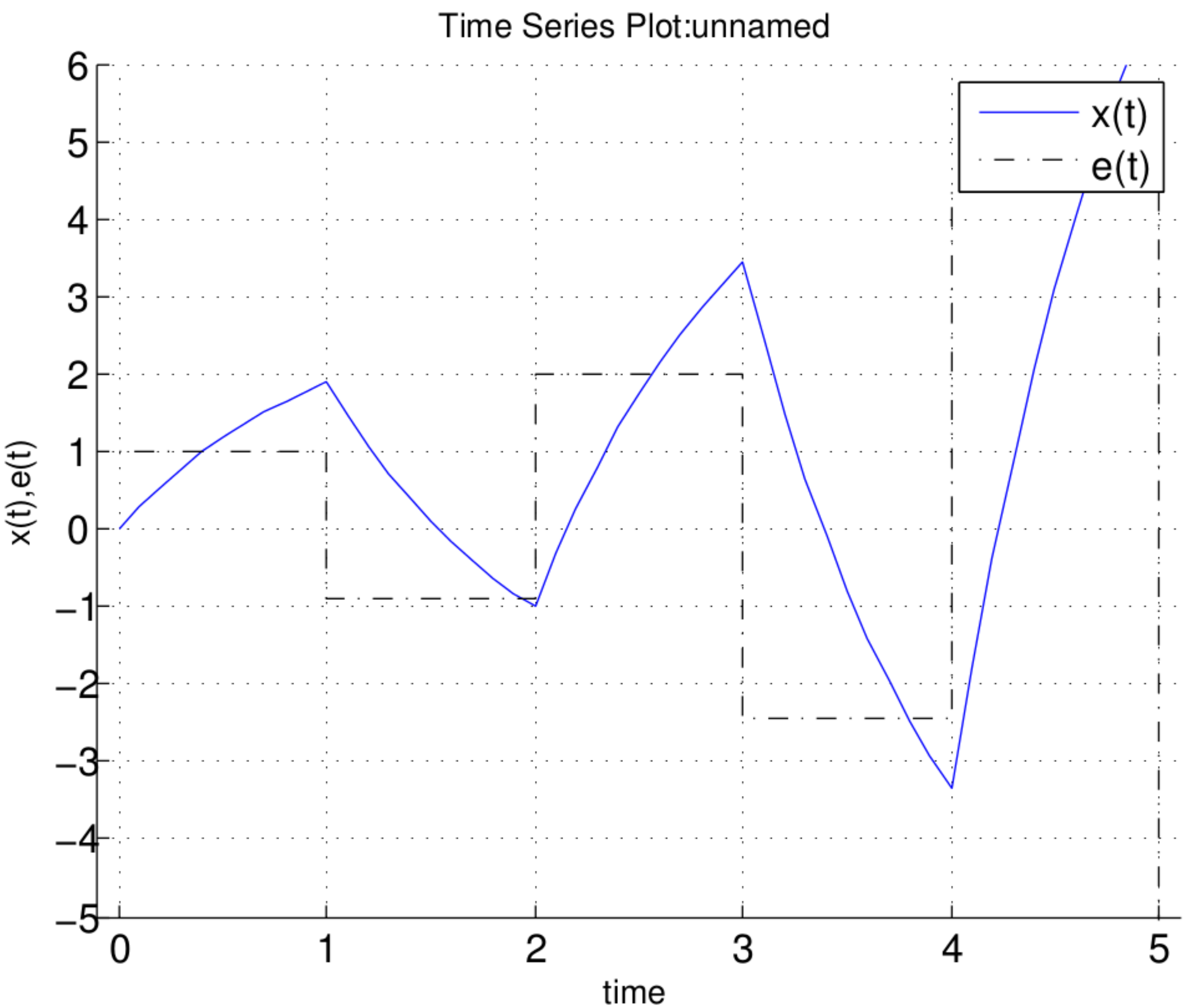
For stability:

$$|e^{aT} - \frac{k_p}{a}(e^{aT} - 1)| < 1. \quad (26)$$

Notes: stability depends on gain **and** T!

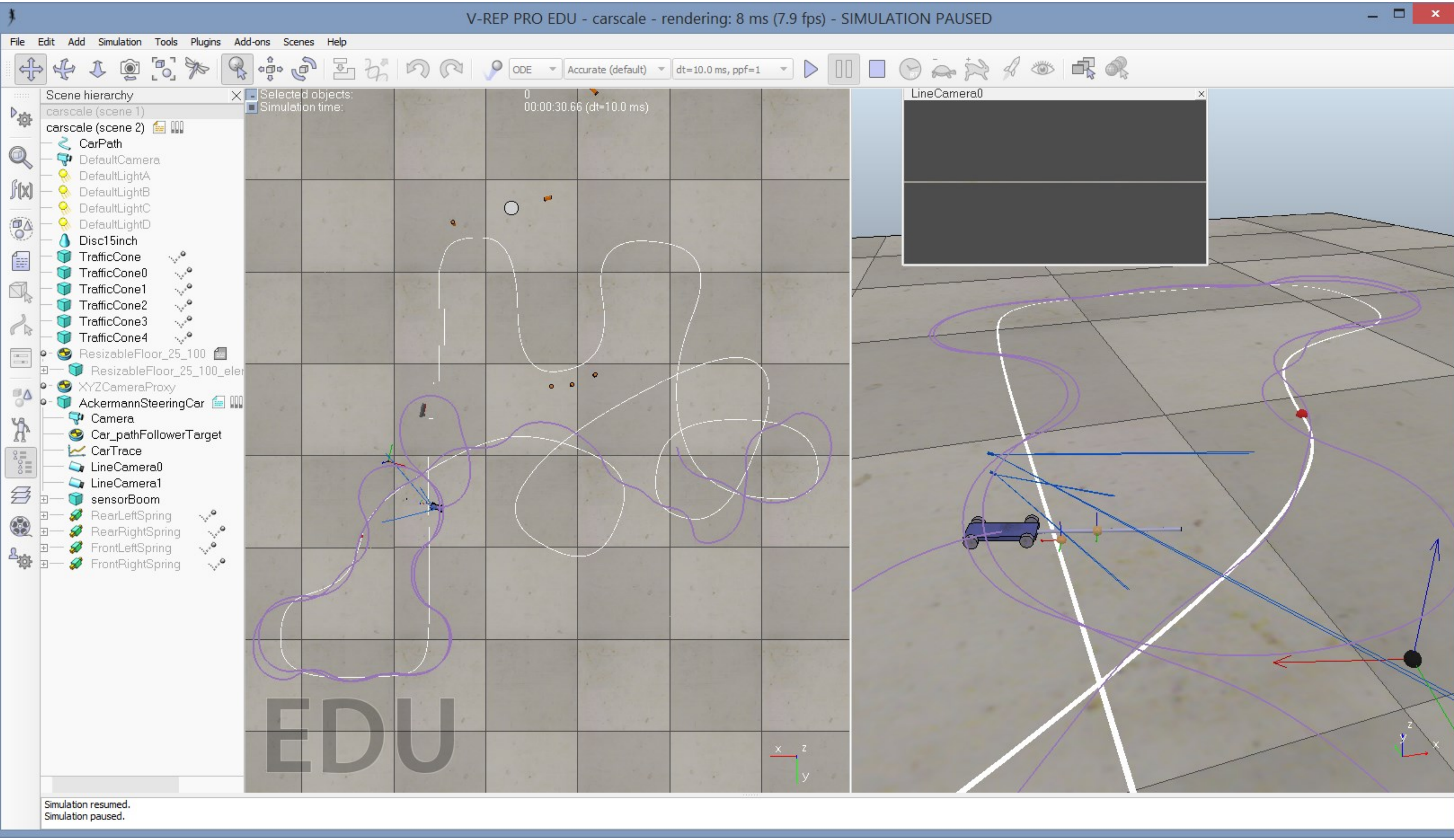
Discrete Time Control

u[k] = kp*(r[k]-x[k])



On board

V-rep simulation



demo

V-rep simulation