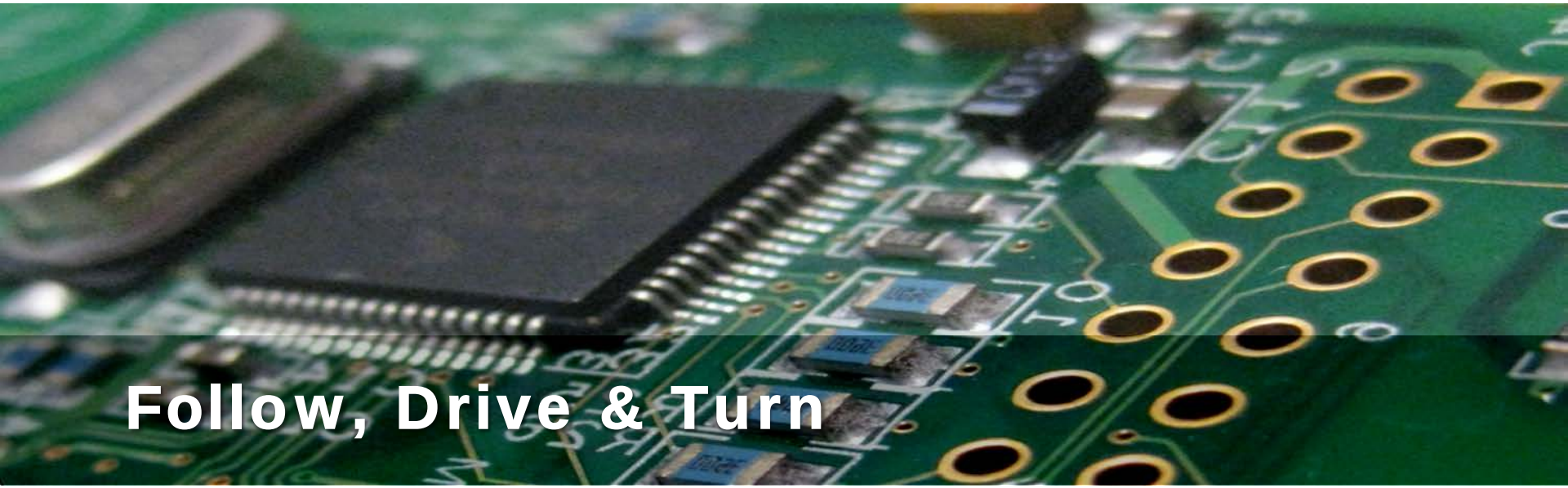
Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



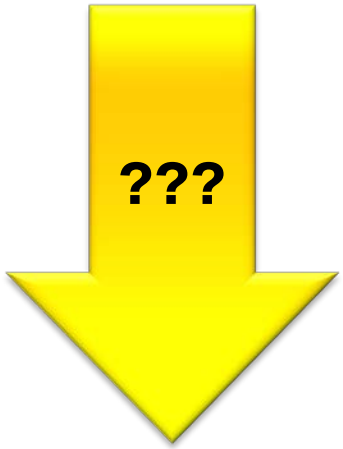
Follow, Drive & Turn

«Follow me, then turn left»

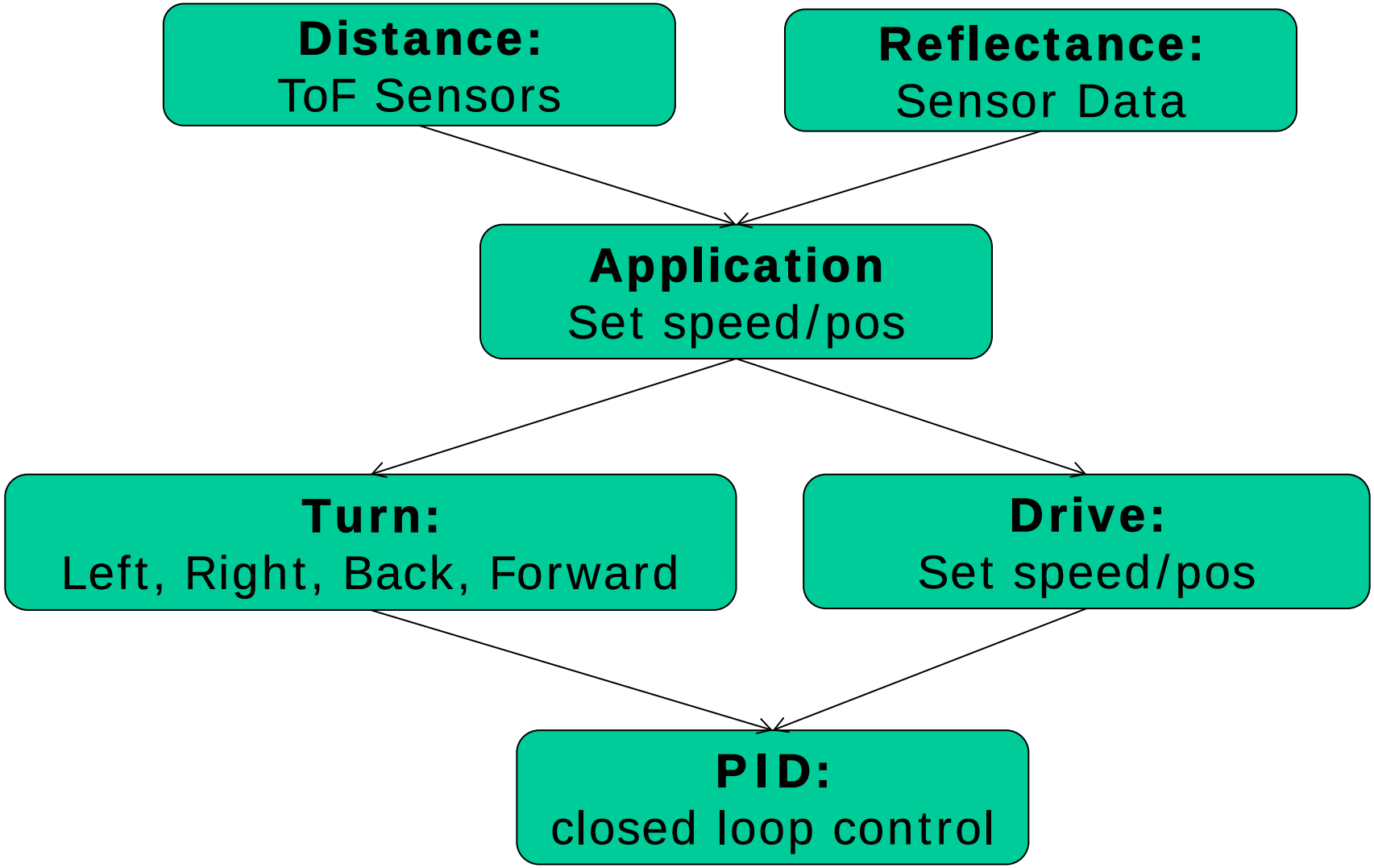
Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Learning Goals

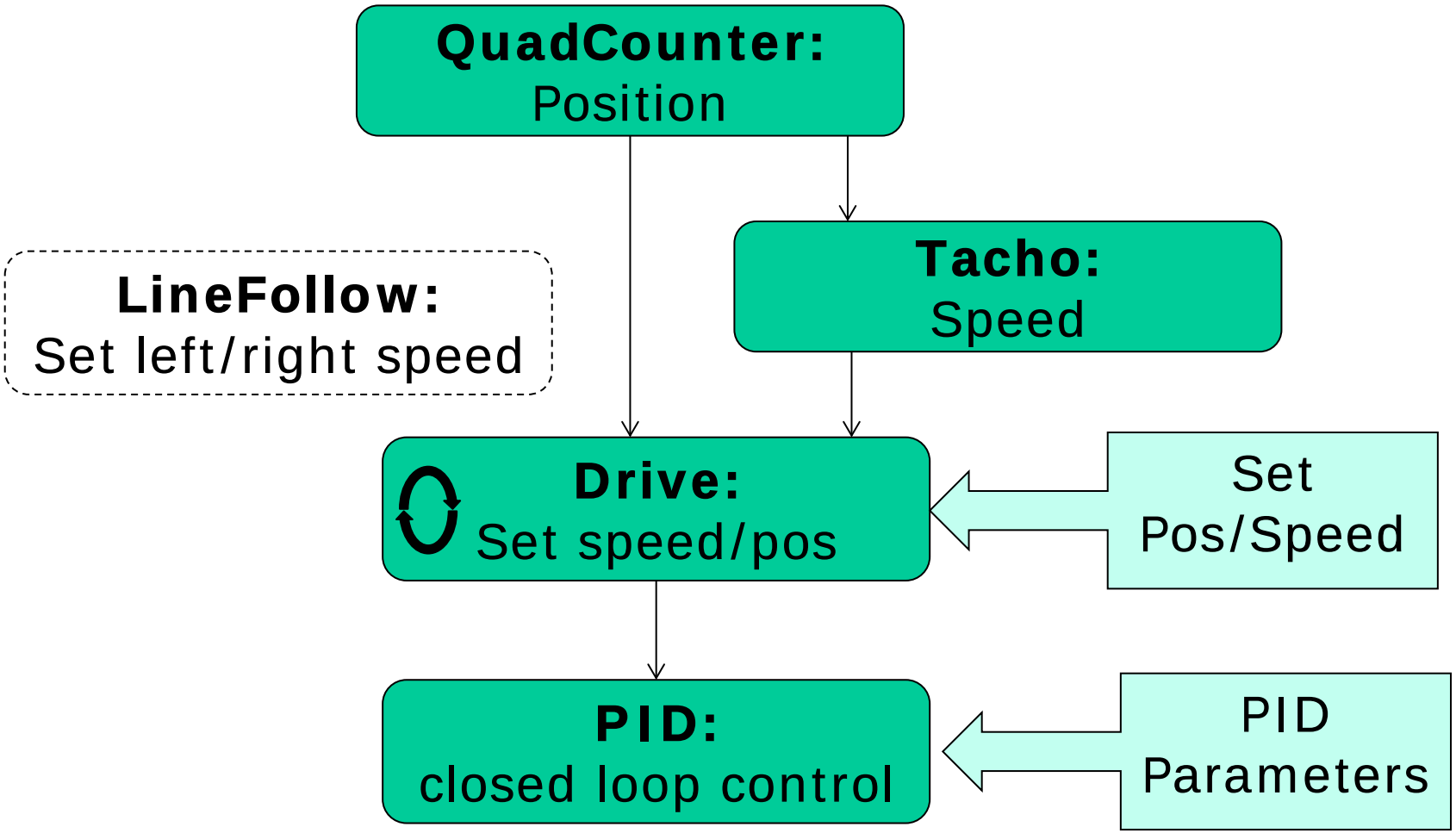
- Using PID to control
  - (Line Following)
  - Position (Turn)
  - Speed (Drive)
- Queues
- FreeRTOS Direct Task Notification



System Overview

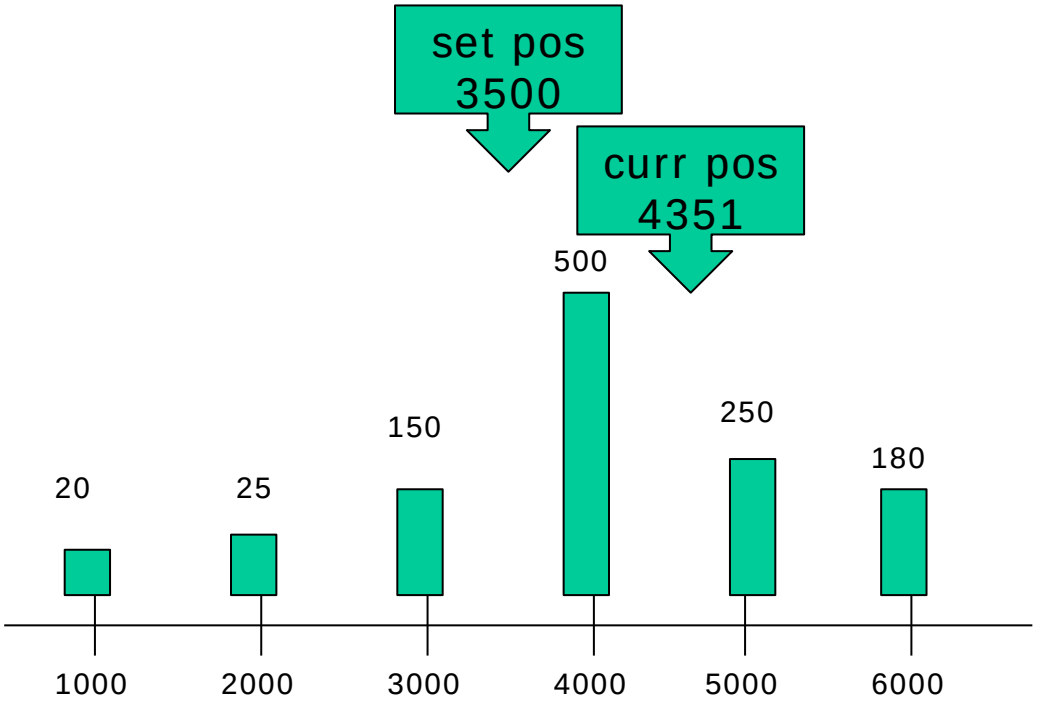


Control Loops



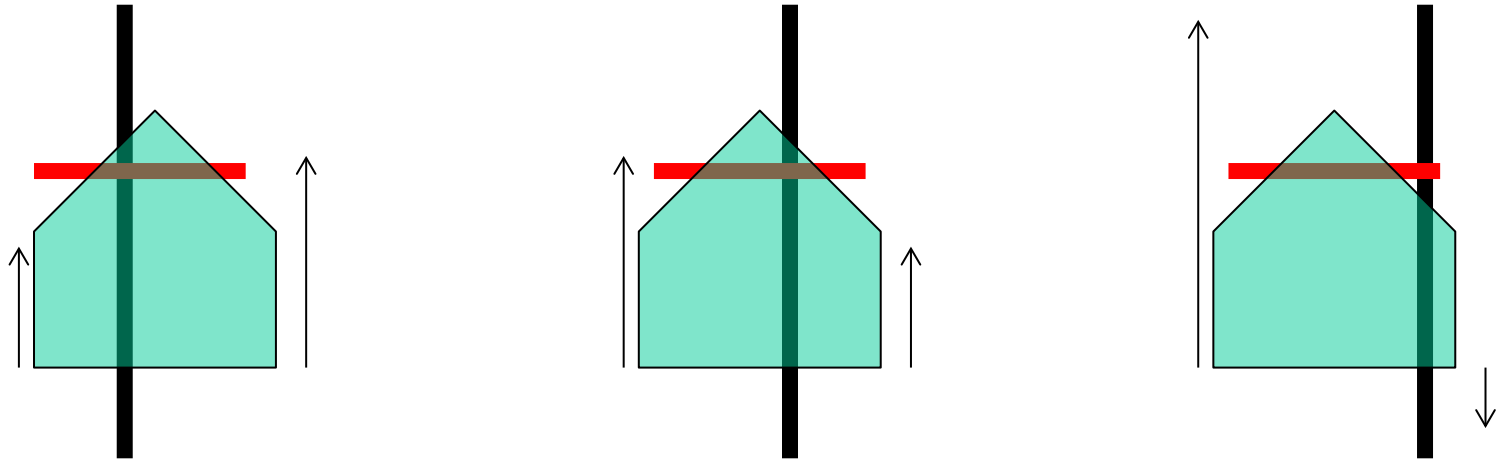
Line Control Loop

- Actual position: REF_GetLineValue()
- PID setpos to middle of sensor

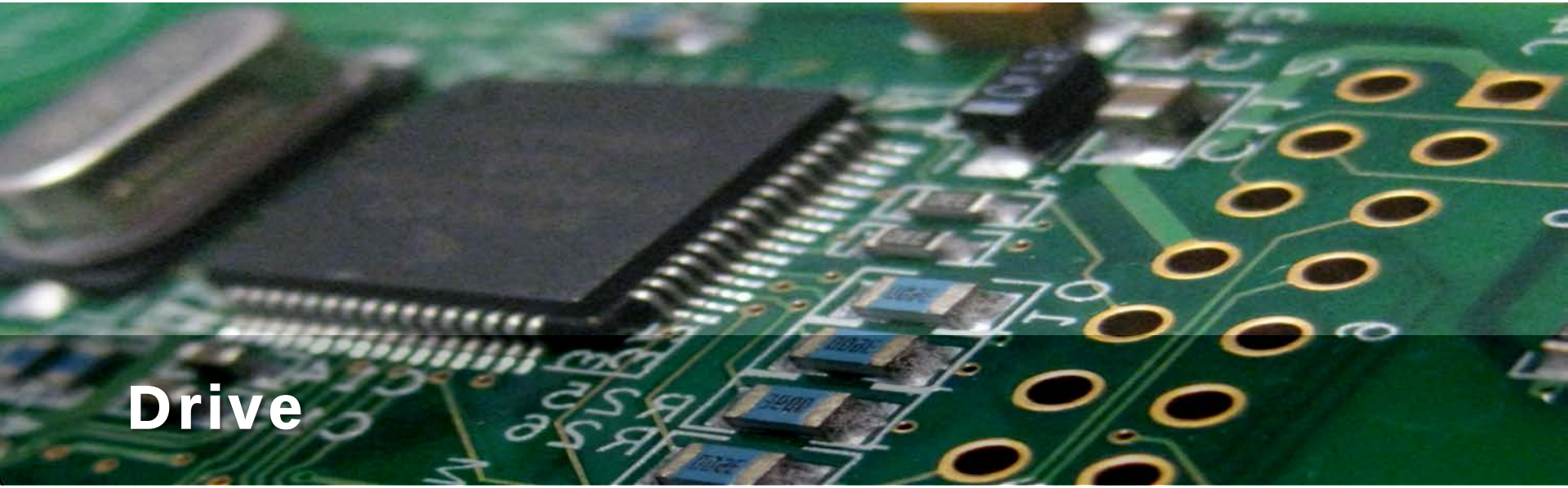


PID Control

- Increase/decrease motor speed depending on PID result
- PID ~ 0: move both motors forward with base speed
- PID >> 0: turn left, left--, right++
- PID << 0: turn right, left++, right --



Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur

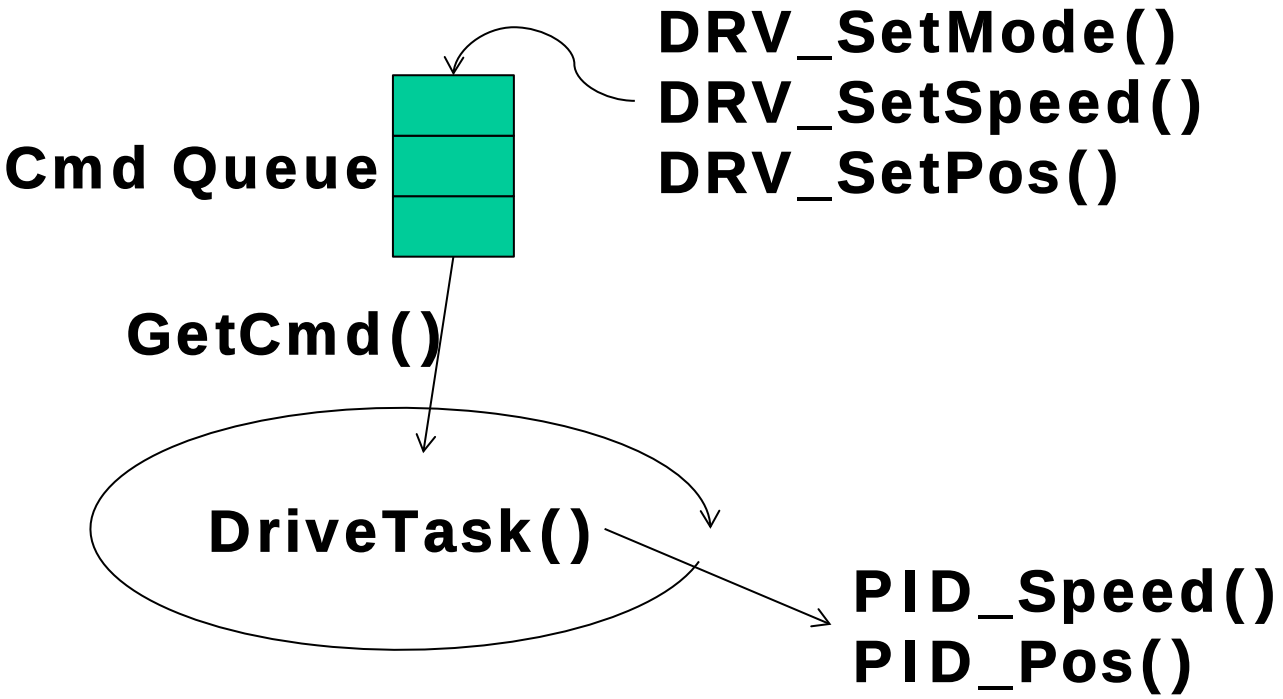


Drive

Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Drive Task

- Drive.c/Drive.h
- Drive Task gets commands and processes them
  - Mode: none, stop, speed, pos
  - Speed: left/right speed
  - Position: left/right position



Drive Shell Commands

```
drive                          ; Group of drive commands
  help|status                  ; Shows drive help or status
  mode <mode>                  ; Set driving mode
                                 (none|stop|speed|pos)
  speed <left> <right>         ; Move left and right motors
                                 with given speed
  pos <left> <right>           ; Move left and right wheels to
                                 given position
  pos reset                    ; Reset drive and wheel position

drive         :
  mode        : NONE
  speed left  : 0 steps/sec (curr: 0)
  speed right: 0 steps/sec (curr: 0)
  pos left    : 0 (curr: 0)
  pos right   : 0 (curr: 0)
```

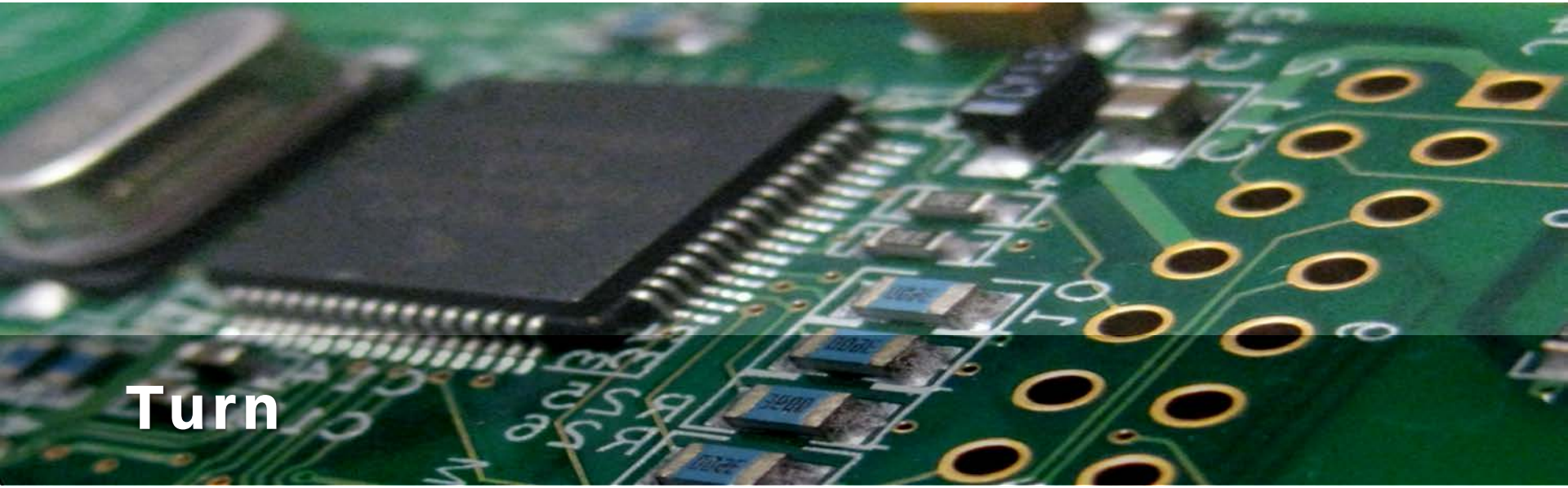
Drive Task

```
static void DriveTask(void *pvParameters) {
  portTickType xLastWakeTime;

  (void)pvParameters;
  xLastWakeTime = xTaskGetTickCount();
  for(;;) {
    while (GetCmd()==ERR_OK) { /* returns ERR_RXEMPTY if queue is empty */
      /* process incoming commands */
    }
    TACHO_CalcSpeed();
    if (DRV_Status.mode==DRV_MODE_SPEED) {
      PID_Speed(TACHO_GetSpeed(TRUE), DRV_Status.speed.left, TRUE);
      PID_Speed(TACHO_GetSpeed(FALSE), DRV_Status.speed.right, FALSE);
    } else if (DRV_Status.mode==DRV_MODE_STOP) {
      PID_Speed(TACHO_GetSpeed(TRUE), 0, TRUE);
      PID_Speed(TACHO_GetSpeed(FALSE), 0, FALSE);
    } else if (DRV_Status.mode==DRV_MODE_POS) {
      PID_Pos(Q4CLeft_GetPos(), DRV_Status.pos.left, TRUE);
      PID_Pos(Q4CRight_GetPos(), DRV_Status.pos.right, FALSE);
    } else if (DRV_Status.mode==DRV_MODE_NONE) {
      /* do nothing */
    }
    vTaskDelayUntil(&xLastWakeTime, 5/portTICK_RATE_MS);
  } /* for */
}
```

Drive Command Queue Handling

```
static uint8_t GetCmd(void) {
  DRV_Command cmd;
  portBASE_TYPE res;

  res = xQueueReceive(DRV_Queue, &cmd, 0);
  if (res==errQUEUE_EMPTY) {
    return ERR_RXEMPTY; /* no command */
  }
  /* process command */
  taskENTER_CRITICAL();
  if (cmd.cmd==DRV_SET_MODE) {
    PID_Start(); /* reset PID, especially integral counters */
    DRV_Status.mode = cmd.u.mode;
  } else if (cmd.cmd==DRV_SET_SPEED) {
    DRV_Status.speed.left = cmd.u.speed.left;
    DRV_Status.speed.right = cmd.u.speed.right;
  } else if (cmd.cmd==DRV_SET_POS) {
    DRV_Status.pos.left = cmd.u.pos.left;
    DRV_Status.pos.right = cmd.u.pos.right;
  }
  taskEXIT_CRITICAL();
  return ERR_OK;
}
```

Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur

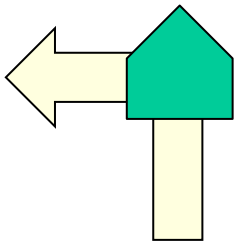


Turn

Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Turning with PID

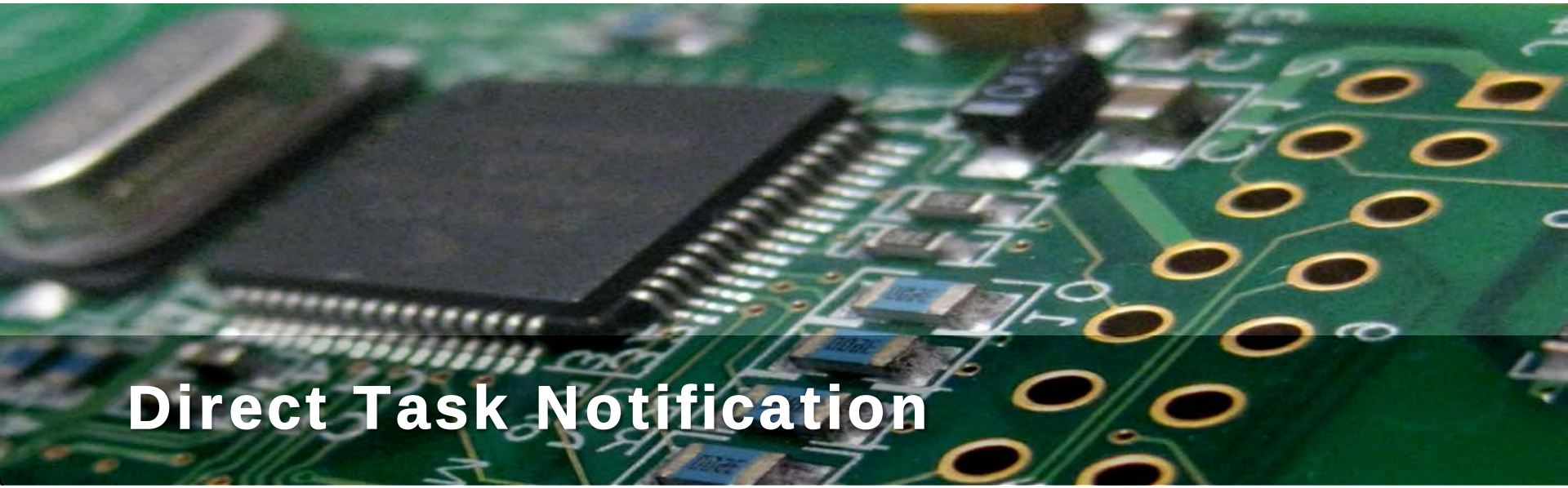
- Left/right by angle, moving forward/backward by steps
- Position-PID
  - Setpoint left wheel position + 50
  - Setpoint right wheel position + 50
  - Run Position PID
- Turn left 90°
  - Setpoint left wheel position + 600
  - Setpoint right wheel position - 600
  - Run Position PID



Turn Shell Commands and Configuration

```
turn                          ; Group of turning commands
  help|status                 ; Shows turn help or status
  <angle>                     ; Turn the robot by angle,
                                negative is counter-clockwise,
                                e.g. 'turn -90'
  forward                     ; Move one step forward
  forward postline            ; Move one step forward post the
                                line
  backward                    ; Move one step backward
  steps90 <steps>             ; Number of steps for a 90 degree
                                turn
  stepsline <steps>           ; Number of steps for stepping
                                over line
  stepspostline <steps>       ; Number of steps for a step post
                                the line
```

Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur

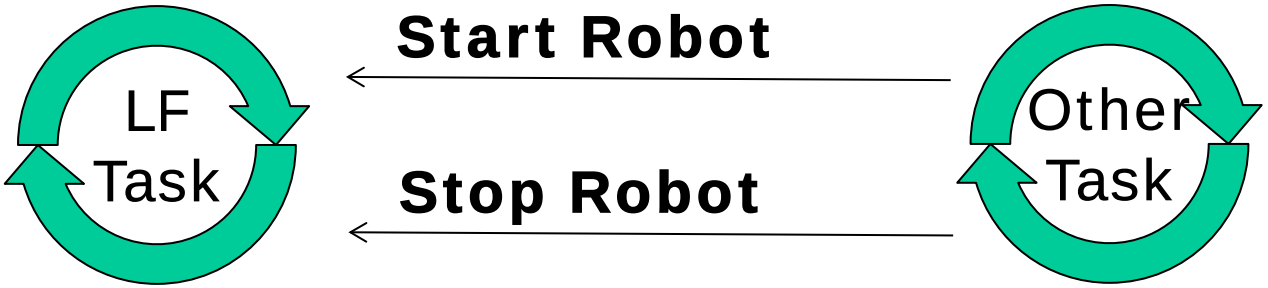


Direct Task Notification

Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

FreeRTOS Direct Task Notification

- Interprocess Communication
  - Semaphore: requires external object
  - Direct Task Notification
- FreeRTOS Direct Task Notification
  - 32bit value stored in TCB plus state (enumeration)
  - No ‘broadcast’: direct, one source, one destination
  - No blocking on send
  - Cannot target ISR
  - No queuing of value (bit set/clear/check)



xTaskNotify (FromISR)

```
typedef enum {
        eNoAction = 0,/* Notify the task without updating its notify value. */
        eSetBits,/* Set bits in the task's notification value. */
        eIncrement,/* Increment the task's notification value. */
        eSetValueWithOverwrite,/* Set the task's notification value to a specific value
even if the previous value has not yet been read by the task. */
        eSetValueWithoutOverwrite /* Set the task's notification value if the previous
value has been read by the task. */
} eNotifyAction;
```

```
BaseType_t xTaskNotify (
        TaskHandle_t xTaskToNotify,
        uint32_t ulValue,
        eNotifyAction eAction);
```

```
BaseType_t xTaskNotifyFromISR(
        TaskHandle_t xTaskToNotify,
        uint32_t ulValue,
        eNotifyAction eAction,
        BaseType_t *pxHigherPriorityTaskWoken);
```

xTaskNotifyWait()

```
BaseType_t xTaskNotifyWait(
        uint32_t ulBitsToClearOnEntry,
        uint32_t ulBitsToClearOnExit,
        uint32_t *pulNotificationValue,
        TickType_t xTicksToWait);
```

- Returns
  - **pdTRUE** if a notification was received, or a notification was already pending when xTaskNotifyWait() was called.
  - **pdFALSE** if the call to xTaskNotifyWait() timed out before a notification was received.

Notification Example

```
#define LF_START_FOLLOWING (1<<0)  /* start line following */
#define LF_STOP_FOLLOWING  (1<<1)  /* stop line following */

static xTaskHandle LFTaskHandle;

void LF_StartFollowing(void) {
  (void)xTaskNotify(LFTaskHandle, LF_START_FOLLOWING, eSetBits);
}


uint32_t notifcationValue;

(void)xTaskNotifyWait(0UL, LF_START_FOLLOWING|LF_STOP_FOLLOWING,
&notifcationValue, 0); /* check flags */
if (notifcationValue&LF_START_FOLLOWING) {
  ...
}
```

Recap ☺

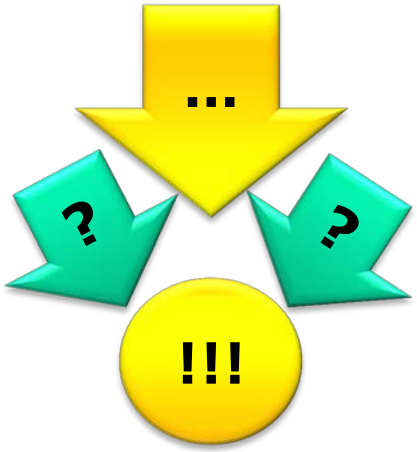
- What is the fundamental difference between **Events** (INTRO BitArray) Events and **FreeRTOS Direct Task Notification** (FDTN)?

- Fill in capability table below:

Capability	Events	FDTN
Notify Interrupt		
Notify multiple tasks		
Block on send		
Block on receive		
Wakeup receiver		

Summary

- PID
  - Turning
  - Driving
  - (Line Following)
- Queues (Drive)
- Direct Task Notification



Lab: Drive and Turn

- Add and integrate
  - Turn.c, Turn.h
  - Drive.c, Drive.h
- Verify PID behavior
  - Drive
    - Speed PID
    - Position PID
  - Turn
    - Left, right, back, forward
- Use **Direct Task Notification** for your own projects
- Ultimate Goal
  - Drive until white border
  - Turn and continue driving