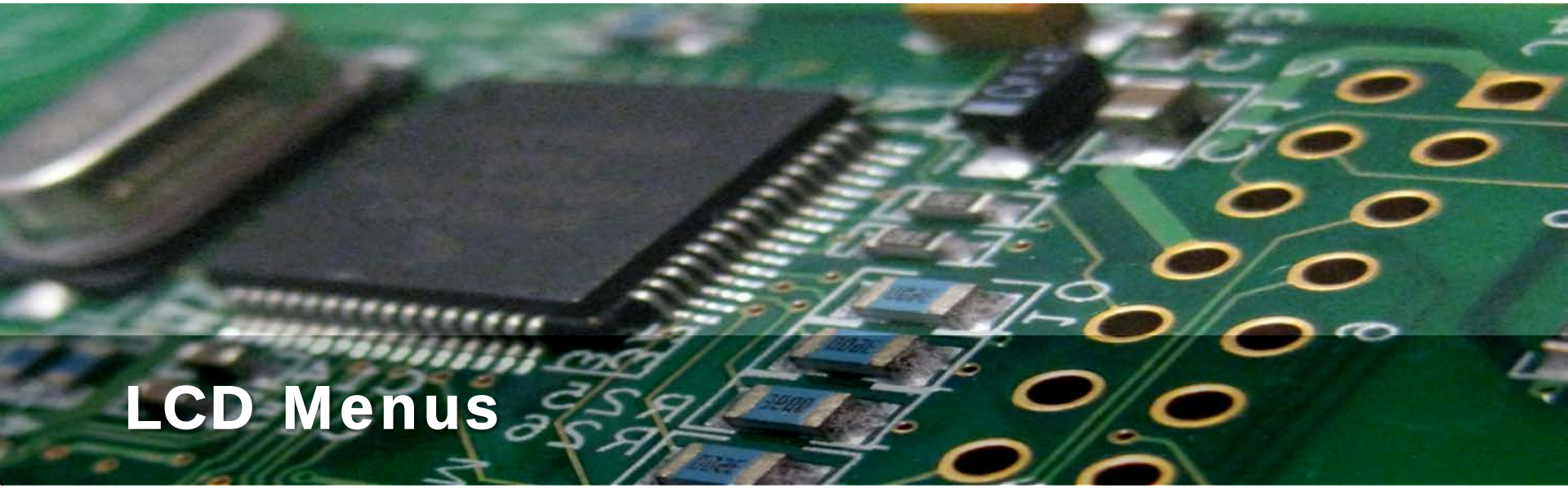
Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



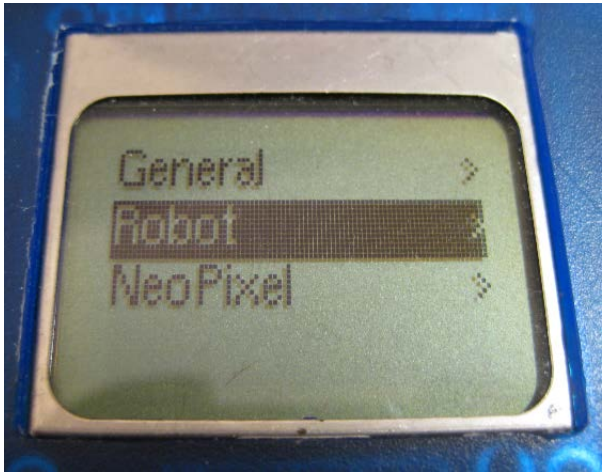
LCD Menus

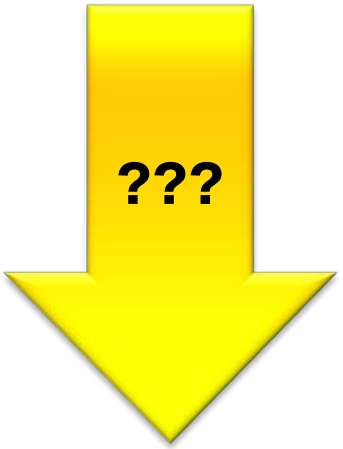
"Mirror, mirror, on the wall, ..."

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

Learning Goals

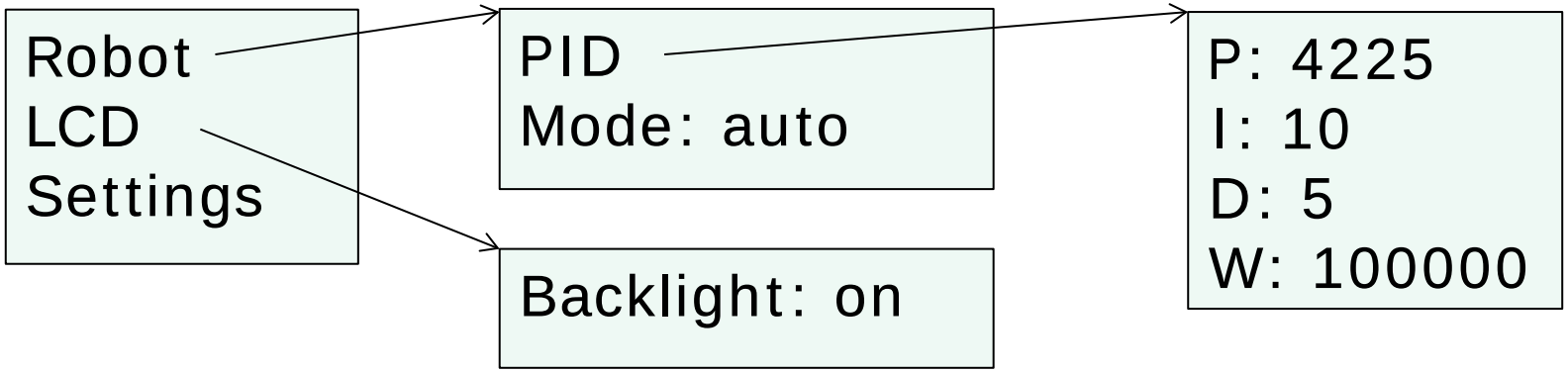
- Text Menus
  - Status
  - Settings
  - Submenu entries
- Events
  - Click, left, right, up, down, ...
  - Draw, Get Text, ...





Hierarchical Menus

- Text Menus
  - Status
  - Settings
  - Submenu entries
- Events
  - Click, left, right, up, down, ...
  - Draw, Get Text, ...



Menu Item Data Structure

- ID: menu ID
- group/pos: hierarchy
- lvlUpID/lvlDownID: upper and sub menu IDs
- menuText: current text
- handler: menu handler for events

```
typedef struct LCDMenu_MenuItem_{
  uint8_t id;  /* unique ID of menu item, starting with 1 */
  uint8_t group; /* menu group, starting with 0 (root), then increasing numbers */
  uint8_t pos; /* position of menu in level, starting with 0 (top position) */
  uint8_t lvlUpID;  /* menu item level up, 0 for 'none' */
  uint8_t lvlDownID; /* menu item level down, 0 for 'none' */
  char *menuText; /* text of menu item */
  bool (*handler)(const struct LCDMenu_MenuItem_ *item,
       LCDMenu_EventType event, void **dataP); /* callback for menu item */
} LCDMenu_MenuItem;
```

0 means no menu

to save ram

use the past data to the menu

MenuID

- Each menu item has an identification
- Special ID for ‘null’ menu ID

```
typedef enum {
  LCD_MENU_ID_NONE = LCDMENU_ID_NONE, /* special value! */  0
  LCD_MENU_ID_MAIN,
  LCD_MENU_ID_BACKLIGHT,
} LCD_MenuIDs;
```

Menu Description

- Constant table with IDs and callbacks
- MenuID: identifies menu item
- Group: Menu Level, starting with 0
- Position inside menu
- Up and Down menu items
- Menu Text (NULL, can be set by handler/callback)
- Manu callback handler
- Flags (for editable items)

```
static const LCDMenu_MenuItem menus[] =
{/* id,               grp, pos, up,             down,                  text,      callback,         flags */
 {LCD_MENU_ID_MAIN,      0, 0, LCD_MENU_ID_NONE, LCD_MENU_ID_BACKLIGHT, "General", NULL,              ...},
 {LCD_MENU_ID_BACKLIGHT, 1, 0, LCD_MENU_ID_MAIN, LCD_MENU_ID_NONE,      NULL,      BackLightMenuHandler,...},
 {LCD_MENU_ID_NUM_VALUE, 1, 1, LCD_MENU_ID_MAIN, LCD_MENU_ID_NONE,      NULL,      ValueChangeHandler,...},
};
```

Menu Events

- Event ID's for event handler
- LCDMenu_OnEvent() sent to menu

```
typedef enum {
  LCDMENU_EVENT_INIT,
  LCDMENU_EVENT_DRAW,
  LCDMENU_EVENT_GET_TEXT, /* get menu text, returned in data handler */
  LCDMENU_EVENT_GET_EDIT_TEXT, /* get menu text in edit mode, returned in data handler */
  LCDMENU_EVENT_UP,
  LCDMENU_EVENT_DOWN,
  LCDMENU_EVENT_LEFT,
  LCDMENU_EVENT_RIGHT,
  LCDMENU_EVENT_ENTER,
  LCDMENU_EVENT_ENTER_EDIT,  /* entering edit mode */
  LCDMENU_EVENT_EXIT_EDIT,   /* exiting edit mode */
  LCDMENU_EVENT_INCREMENT,
  LCDMENU_EVENT_DECREMENT
} LCDMenu_EventType;

void LCDMenu_OnEvent(LCDMenu_EventType event);
```

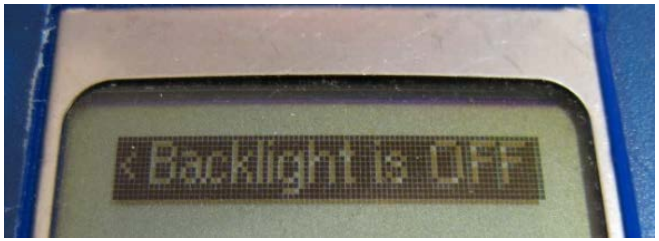
Event Handler

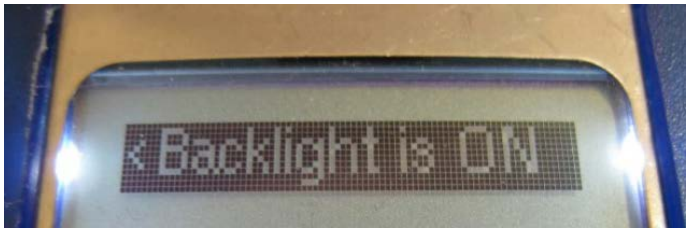
- Reacts on event for menu ID

Callback BackLightMenuHandler

```
static LCDMenu_StatusFlags BackLightMenuHandler(
  const struct LCDMenu_MenuItem_ *item,
  LCDMenu_EventType event, void **dataP)
{
  LCDMenu_StatusFlags flags = LCDMENU_STATUS_FLAGS_NONE;

  if (event==LCDMENU_EVENT_GET_TEXT && dataP!=NULL) {
    if (LedBackLightisOn) {
      *dataP = "Backlight is ON";
    } else {
      *dataP = "Backlight is OFF";
    }
    flags |= LCDMENU_STATUS_FLAGS_HANDLED|LCDMENU_STATUS_FLAGS_UPDATE_VIEW;
  } else if (event==LCDMENU_EVENT_ENTER) { /* toggle setting */
    LedBackLightisOn = !LedBackLightisOn;
    flags |= LCDMENU_STATUS_FLAGS_HANDLED|LCDMENU_STATUS_FLAGS_UPDATE_VIEW;
  }  return flags;
}
```

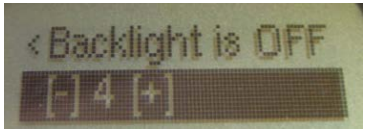




Event Handler to Change Item

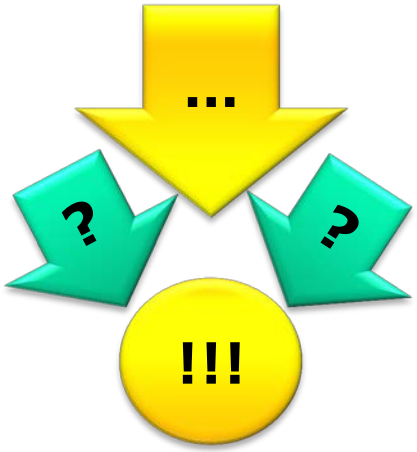
```
static LCDMenu_StatusFlags ValueChangeHandler(const struct LCDMenu_MenuItem_ *item,
    LCDMenu_EventType event, void **dataP)
{
  static int value = 0;
  static uint8_t valueBuf[16];
  LCDMenu_StatusFlags flags = LCDMENU_STATUS_FLAGS_NONE;

  (void)item;
  if (event==LCDMENU_EVENT_GET_TEXT) {
    UTIL1_strcpy(valueBuf, sizeof(valueBuf), (uint8_t*)"Val: ");
    UTIL1_strcatNum32s(valueBuf, sizeof(valueBuf), value);
    *dataP = valueBuf;
    flags |= LCDMENU_STATUS_FLAGS_HANDLED|LCDMENU_STATUS_FLAGS_UPDATE_VIEW;
  } else if (event==LCDMENU_EVENT_GET_EDIT_TEXT) {
    UTIL1_strcpy(valueBuf, sizeof(valueBuf), (uint8_t*)"[-] ");
    UTIL1_strcatNum32s(valueBuf, sizeof(valueBuf), value);
    UTIL1_strcat(valueBuf, sizeof(valueBuf), (uint8_t*)" [+]");
    *dataP = valueBuf;
    flags |= LCDMENU_STATUS_FLAGS_HANDLED|LCDMENU_STATUS_FLAGS_UPDATE_VIEW;
  } else if (event==LCDMENU_EVENT_DECREMENT) {
    value--;
    flags |= LCDMENU_STATUS_FLAGS_HANDLED|LCDMENU_STATUS_FLAGS_UPDATE_VIEW;
  } else if (event==LCDMENU_EVENT_INCREMENT) {
    value++;
    flags |= LCDMENU_STATUS_FLAGS_HANDLED|LCDMENU_STATUS_FLAGS_UPDATE_VIEW;
  }
  return flags;
# 9 }
```



Summary

- Menu structure described in table
- IDs to identify menu items
- Hierarchical menu
  - Level, Position, Parent and Children
- Events sent to menu
  - Get menu text
  - Change menu text
  - Change value



Lab: LCD Menu

- Enable LCD menu module
- Route push button events to LCD menu
- Extend menus
  - Changing values
  - Display values from the robot
  - Changing values on the robot