These slides serve as a visual aid for the lecture, not as a comprehensive document or script.

Please refrain from printing these slides to help protect the environment.

For any comments or feedback, please contact n.meseth@hs-osnabrueck.de.

Programming Challenges

Exercises for the book
Hands-On Computer Science

with the LED

boilerplate code

```python
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_rgb_led_v2 import BrickletRGBLEDV2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
led = BrickletRGBLEDV2("<YOUR_UID>", ipcon)
led.set_rgb_value(0, 0, 0)
```

challenge #1 - color selection ✅

Ask the user to enter a color name (“red”, “green”, “blue”). The LED should light up in that color. If the input is unknown, the LED should turn white.

challenge #2 - brightness control ✅

```
Ask the user to enter a number between 0 and 100. The LED
should light up with the corresponding brightness.
```

challenge #3 - on/off control ✅

Continuously ask the user whether the LED should be turned “on” or “off”. The program ends only when the user enters “stop”.

challenge #4 - blinking sequence ✅

Ask the user for a number. The LED should blink exactly that many times and then stay off.

Add-on: Let the user choose the speed of the blinking, too.

challenge #5 - traffic light ✅

Simulate a traffic light by cycling through red, yellow, and green automatically, each color for a short time.

challenge #6 - random color ✅

```
Every time the user presses Enter, the LED should show a
randomly chosen color.
```

challenge #7 - morse code ❌

The LED blinks the word “SOS” in Morse code using short and long flashes.

with the rotary encoder

challenge #8 - counter ✅

Turning the knob changes a counter value. The program prints the current value regularly in the console.

Add-on: the counter can only move between 0 and 10. If the knob is turned beyond this range, the value stays at the limit. ✅

Add-on: a short press resets the counter to 0. A long press shows a message in the console. ❌

challenge #9 - direction indicator ✅

```
The program detects whether the knob is turned to the left
or to the right and prints the direction.
```

challenge #10 - rotation speed level 1 ❌

The program measures how fast the knob is turned and indicates whether it is slow or fast by printing "slow" or "fast" to the console. No rotation → print "still".

challenge #11 - color menu level 1 ✅

Turning the knob scrolls through several color options. Pressing the knob confirms the current selection and prints it to the console.

challenge #12 - speedometer ❌

`The program counts how many steps the knob is turned in one second and prints the number continuously. How fast can you go?`

with LED AND the rotary encoder

challenge #13 - rotation speed level 2 ❌

```
The program measures how fast the knob is turned and
indicates whether it is slow or fast by lighting up the LED
from green (slow) over orange (medium) to red (fast). No
rotation → LED is off.
```

challenge #14 - color menu level 2 ❌

```
Turning the knob scrolls through several color options.
Pressing the knob confirms the current selection and lights
up the LED in the corresponding color.
```

with IR sensor

boilerplate code

```python
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_distance_ir_v2 import BrickletDistanceIRV2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
ir = BrickletDistanceIRV2("<YOUR_UID>", ipcon)
```

challenge #15 - read & print ✅

```
Read the distance value once per second and print it to the
console.

Add-on: Let the user enter the time between measurements on
program start.
```

challenge #16 - min/max tracker ✅

Continuously measure distance and keep track of the minimum and maximum seen so far. Print them every 5 seconds.

challenge #17 - wall detector ✅

If distance is below a threshold (e.g., 150 mm), print “TOO CLOSE!”, else print “OK”.

Add-on: Let the user enter the threshold on program start.

challenge #18 - obstacle detector ✅

```
Continuously classify the measured distance into 3 zones
"near", "medium", "far", and print the zone name when it
changes.
```

challenge #19 - smoothing ❌

Continuously store the last 5 measurements in a list and display the average instead of the last raw value.

Add-on: Let the user enter the number of measurements used for average calculation.

challenge #20 - direction detection ✅

`Continuously compare the current distance to the previous distance and print “approaching”, “moving away”, or “stable” (with a small tolerance, e.g., ±5 mm).`

with OLED display

boilerplate code

```python
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_oled_128x64_v2 import BrickletOLED128x64V2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
oled = BrickletOLED128x64V2("<YOUR_UID>", ipcon)

oled.clear_display()
```

challenge #21 - greetings ✅

Ask the user for his name and print it a greeting to the display.

Add-On: Include the current year in a greeting like "Welcome {name} in the year {year}" ❌

challenge #22 - counter ✅

`Show a counter from 0 to 100, updating every 0.1s.`

`Add-on: Count down instead.`

challenge #23 - digital clock ✅

`Display the current time (HH:MM:SS) and update once per second.`

challenge #24 - short jokes ❌

Store 5 short 2-line jokes in a list. Each time the user presses Enter, show a random one on the OLED.

Add-on: Let ChatGPT generate a new joke every time instead of storing a set of jokes beforehand.

challenge #25 - progress bar ❌

Draw a progress bar from 0% to 100% over 10 seconds.

Add-on: Show the numeric percentage next to it.

challenge #26 - scrolling text ❌

Display a long sentence that scrolls from right to left across the screen.

Hint: shift x-position or show substring windows.

challenge #27 - drawing ❌

In a loop, let the user enter either "up", "down", "left", or "right". Draw 5 pixels on the display in the chosen direction. Next time, start from where the pen stopped.