

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

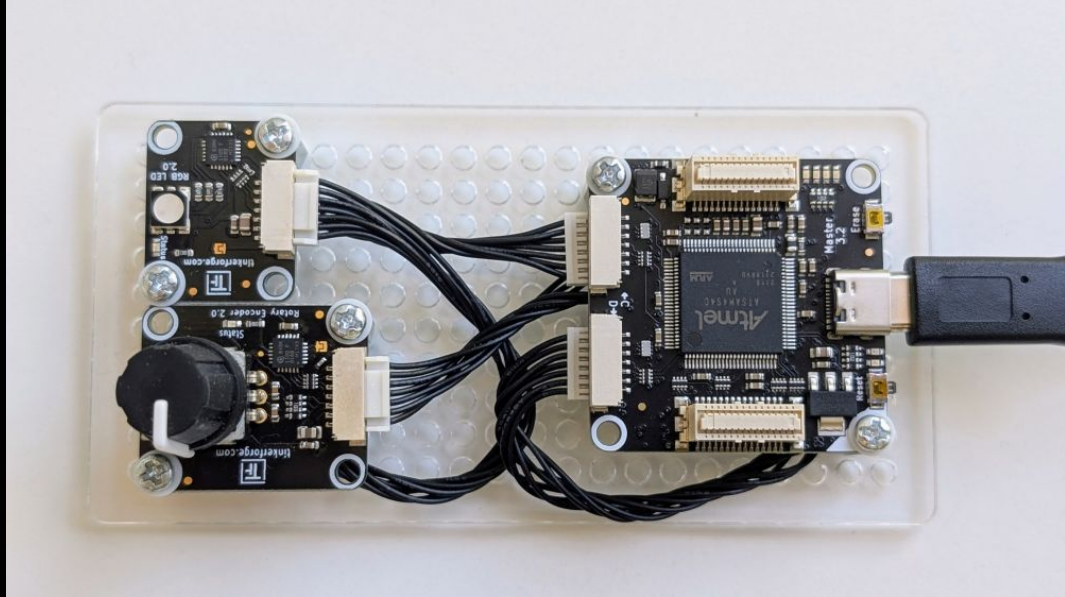
Please refrain from printing these slides to help protect the environment.

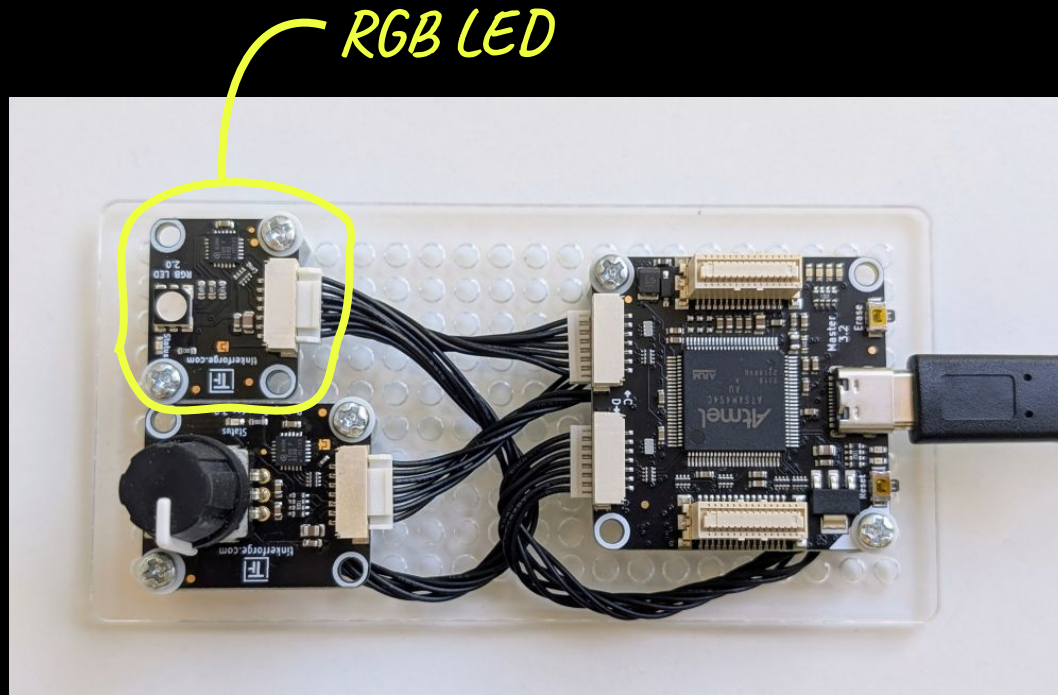
For any comments or feedback, please contact n.meseth@hs-osnabrueck.de.

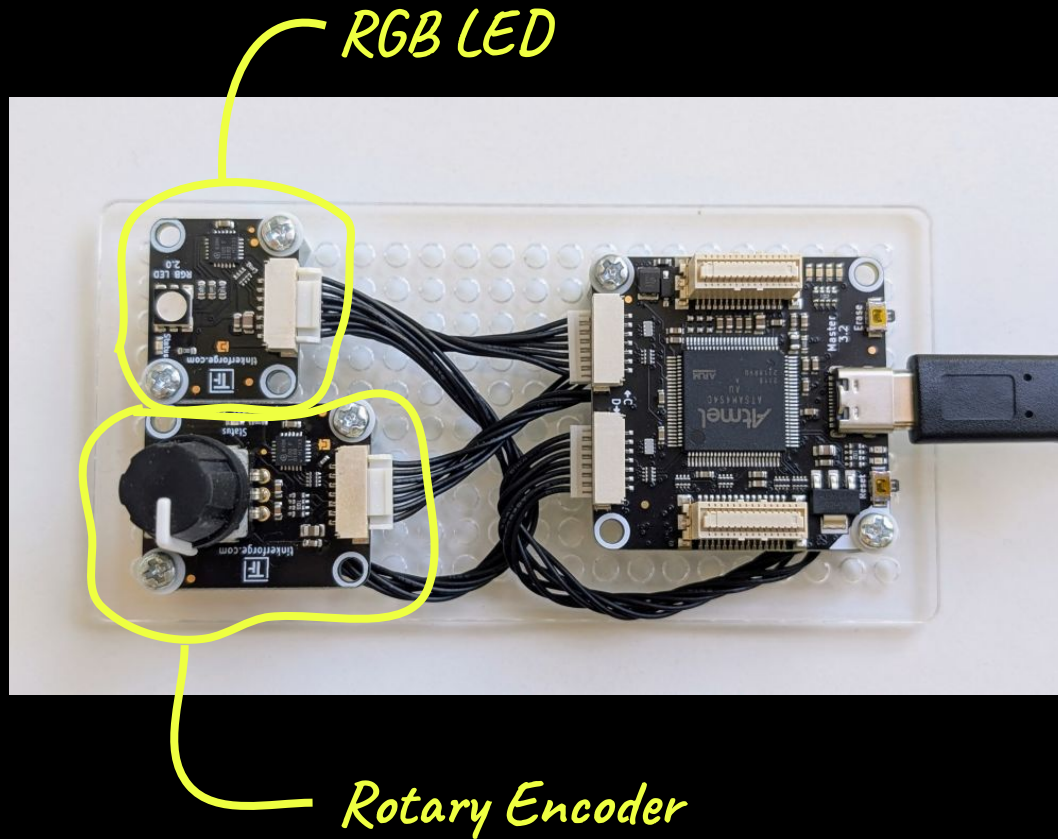


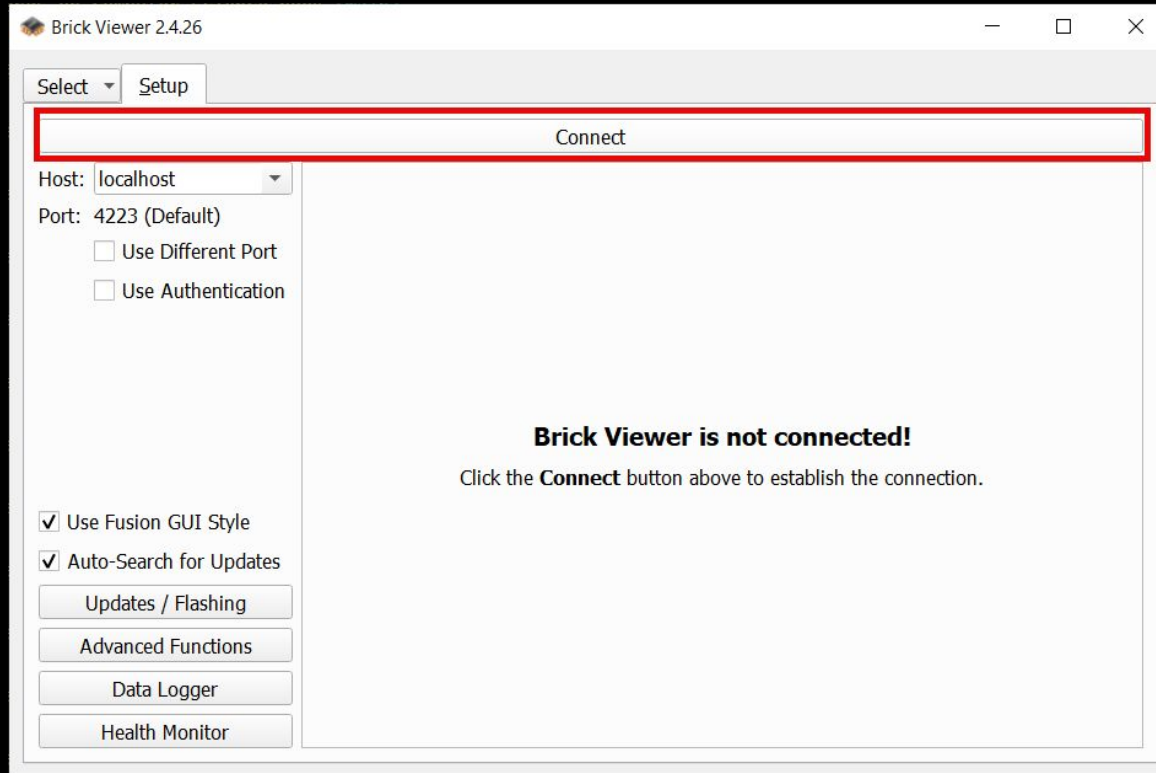
IMAGES

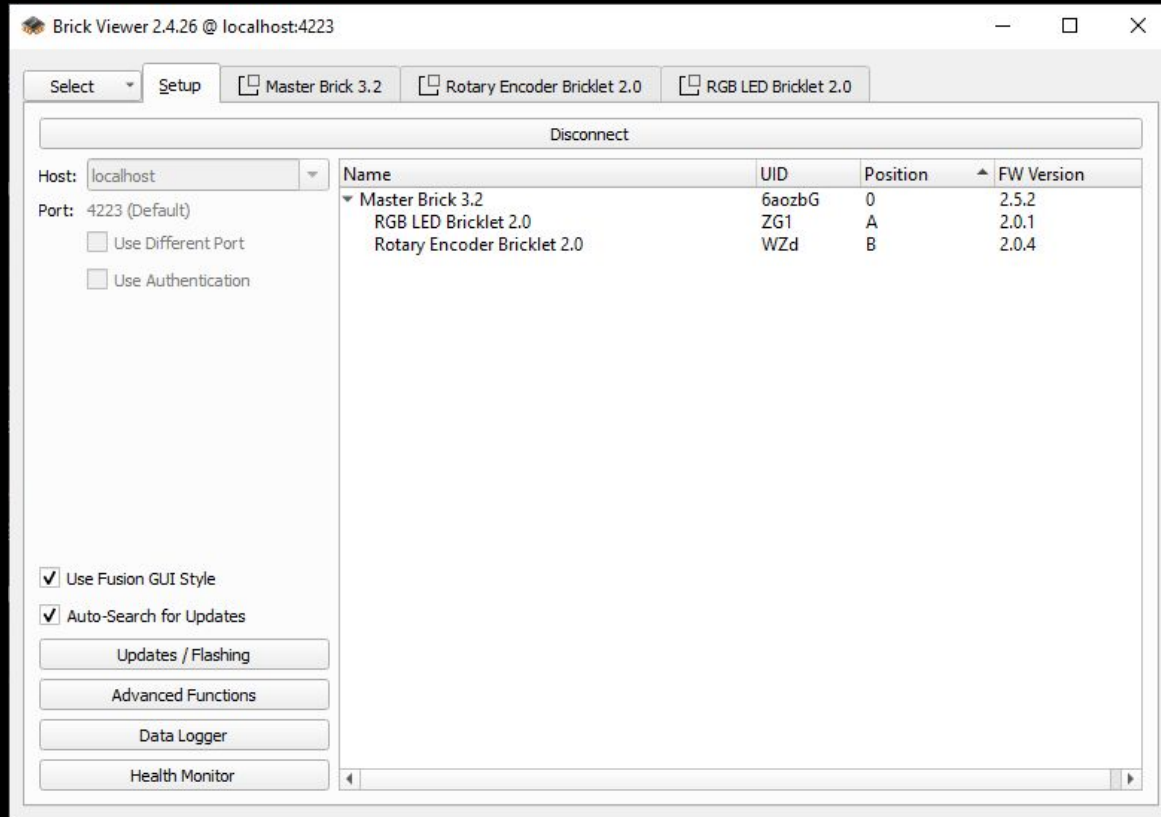
Supporting slides for chapter 4 of the book
Hands-On Computer Science

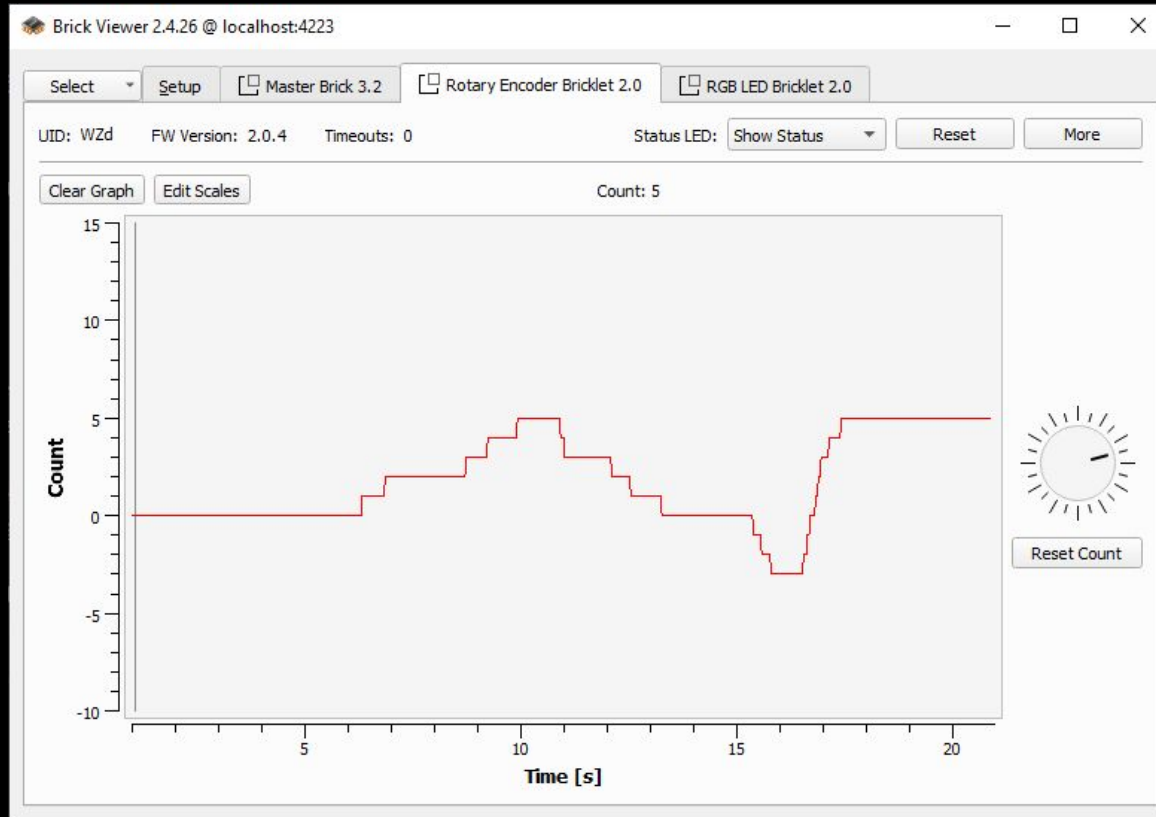












boilerplate code

```
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_rotary_encoder_v2 import BrickletRotaryEncoderV2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
knob = BrickletRotaryEncoderV2("WZd", ipcon)
```

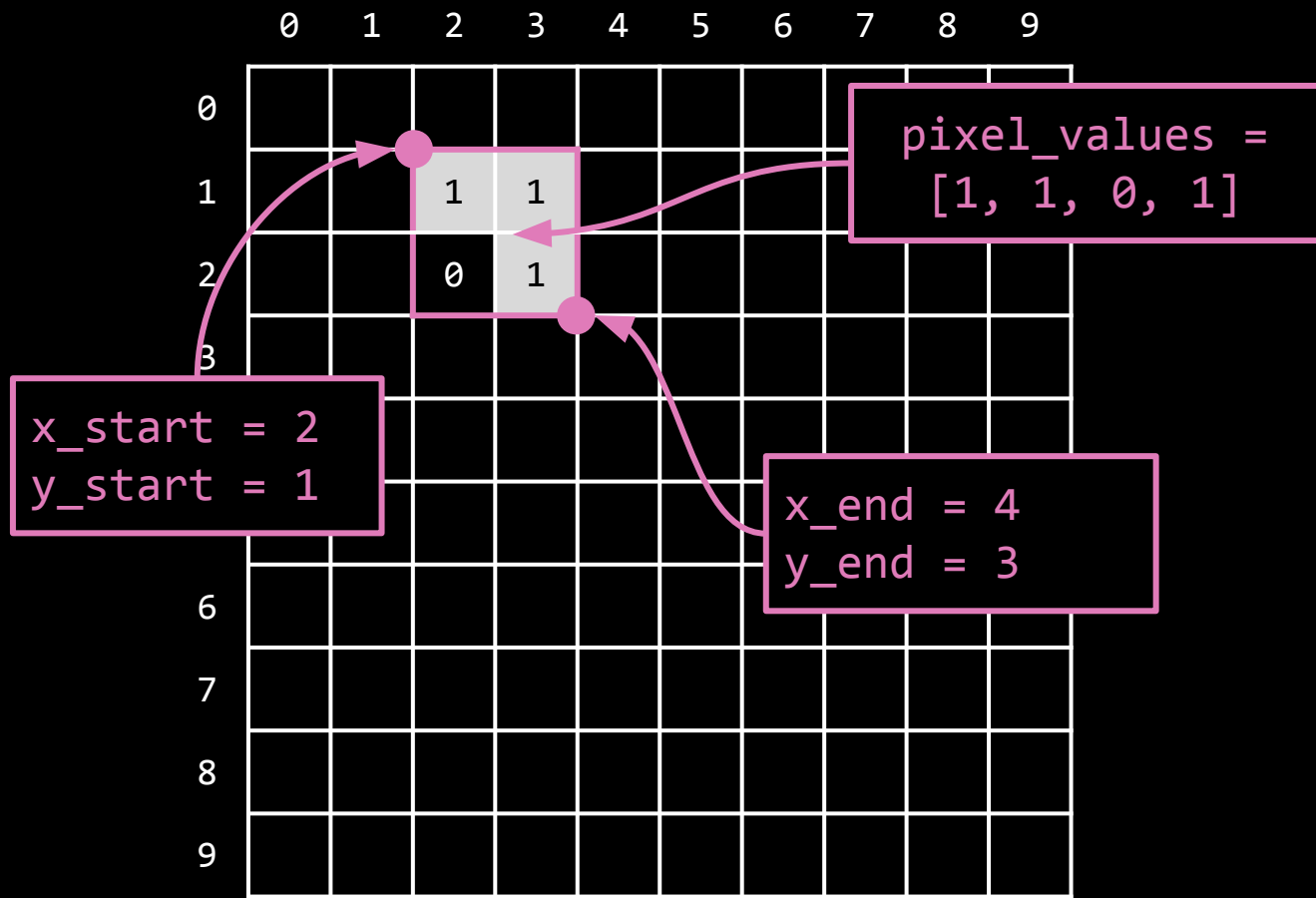
reading the counter

```
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_rotary_encoder_v2 import BrickletRotaryEncoderV2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
knob = BrickletRotaryEncoderV2("WZd", ipcon)

count = knob.get_count(reset=False)
```

PIXELS



CONTROL STRUCTURES

keyword

```
if new_count != last_count:  
    last_count = new_count  
    print(last_count)
```

keyword followed by a *condition (true/false)*

```
if new_count != last_count:  
    last_count = new_count  
    print(last_count)
```

keyword followed by a *condition (true/false)*

```
if new_count != last_count:  
    last_count = new_count  
    print(last_count)
```

*this code runs only if
condition is true!*

LED DIMMER V1

```
knob.reset()
last_count = 0

while True:
    new_count = knob.get_count(reset=False)

    if new_count != last_count:
        last_count = new_count
        led.set_rgb_value(last_count, last_count, last_count)
```

```
knob.reset()
last_count = 0

while True:
    new_count = knob.get_count(reset=False)

    if new_count != last_count:
        last_count = new_count
        led.set_rgb_value(last_count, last_count, last_count)
```

struct.error: ubyte format requires 0 <= number <= 255

NUMBER SYSTEMS

BITS & BYTES

LED DIMMER V2

```
brightness = 0
```

```
...
```

```
diff = new_count - last_count
```

```
brightness += diff
```

```
brightness = max(0, min(255, brightness))
```


definition of a constant

STEP = 10

...

brightness += diff * STEP

brightness = max(0, min(255, brightness))

READING THE BUTTON

```
while True:
    if knob.is_pressed():
        print("Button pressed")
    else:
        print("Button not pressed")
```

LED DIMMER V3

```
if color == "white":  
    led.set_rgb_value(brightness, brightness, brightness)  
if color == "yellow":  
    led.set_rgb_value(brightness, brightness, 0)  
if color == "green":  
    led.set_rgb_value(0, brightness, 0)
```

```
button_pressed_before = False
while True:
    button_pressed_after = knob.is_pressed()

    if button_pressed_before == True and button_pressed_after == False:
        if color == "white":
            color = "yellow"
        elif color == "yellow":
            color = "green"
        elif color == "green":
            color = "white"

    button_pressed_before = button_pressed_after
```

FUNCTIONS

defining a function

```
def set_led_color(color, brightness):  
    if color == "white":  
        led.set_rgb_value(brightness, brightness, brightness)  
    if color == "yellow":  
        led.set_rgb_value(brightness, brightness, 0)  
    if color == "green":  
        led.set_rgb_value(0, brightness, 0)
```


defining a function with a chosen name

```
def set_led_color(color, brightness):  
    if color == "white":  
        led.set_rgb_value(brightness, brightness, brightness)  
    if color == "yellow":  
        led.set_rgb_value(brightness, brightness, 0)  
    if color == "green":  
        led.set_rgb_value(0, brightness, 0)
```

defining a function with a chosen name and parameters

```
def set_led_color(color, brightness):  
    if color == "white":  
        led.set_rgb_value(brightness, brightness, brightness)  
    if color == "yellow":  
        led.set_rgb_value(brightness, brightness, 0)  
    if color == "green":  
        led.set_rgb_value(0, brightness, 0)
```