

0. ORGANIZATION
1. DIGITAL TECHNOLOGIES
2. SENSORS
3. ACTUATORS
4. COMPUTER VISION
5. GENERATIVE AI
6. NATURAL LANGUAGE PROCESSING
7. USER INTERFACES
8. CLOUD SERVICES
9. DATABASES

The slides are meant as visual support for the lecture.
They are neither a documentation nor a script.

Please do not print the slides.

Comments and feedback at n.meseth@hs-osnabrueck.de

ORGANIZATION

ILIAS

Microsoft Teams

sessions

group work

examination

working environment

visual studio code
python
tinkerforge
git

DIGITAL TECHNOLOGIES

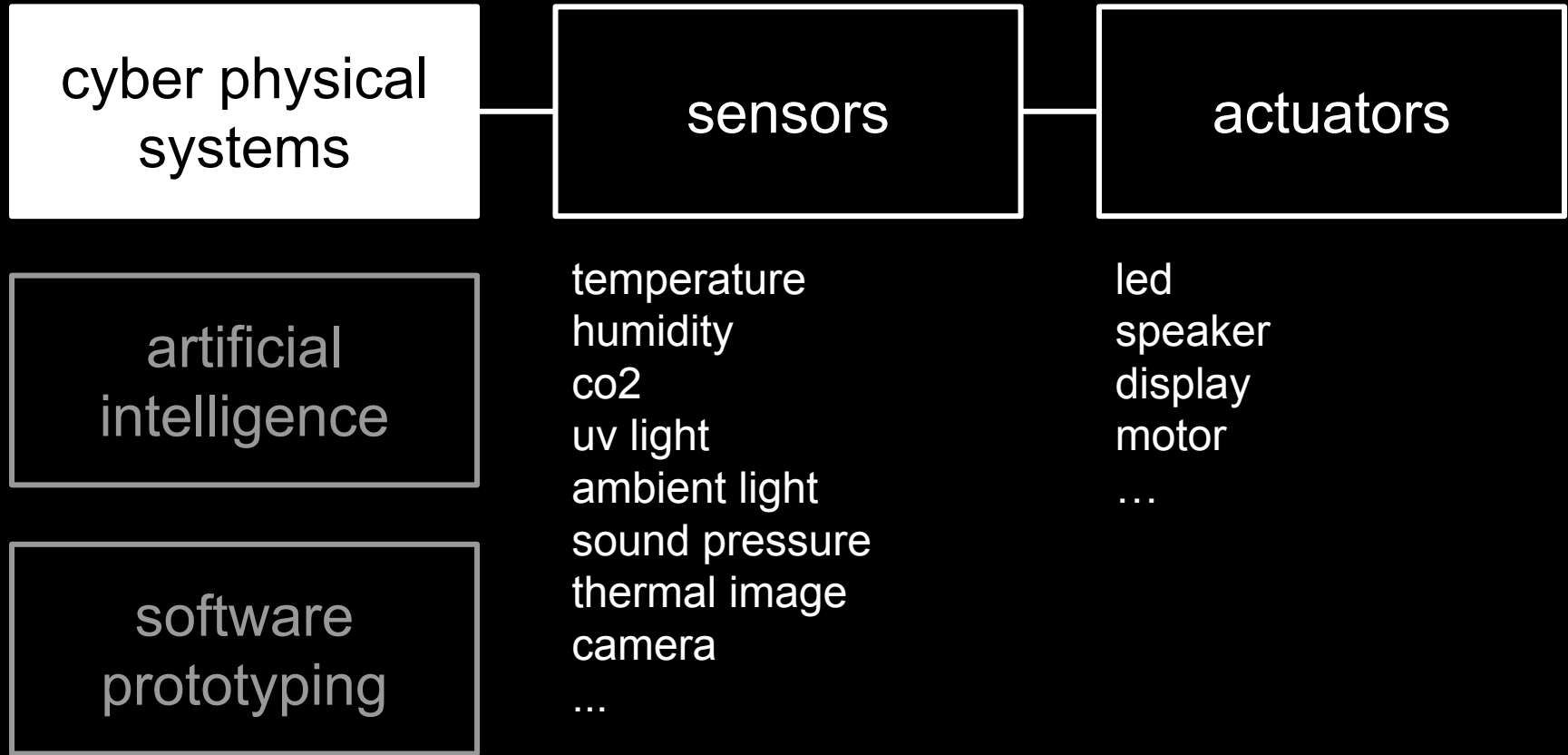
a model for solving problems

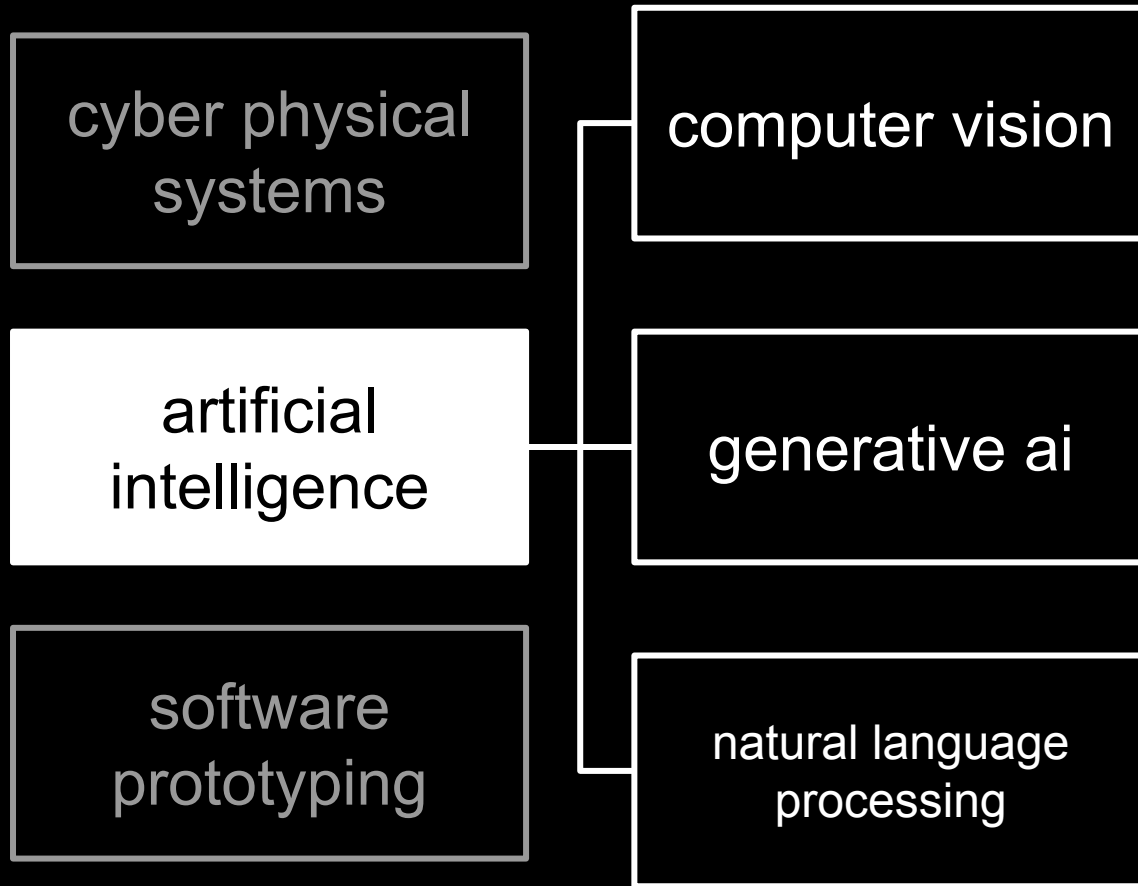


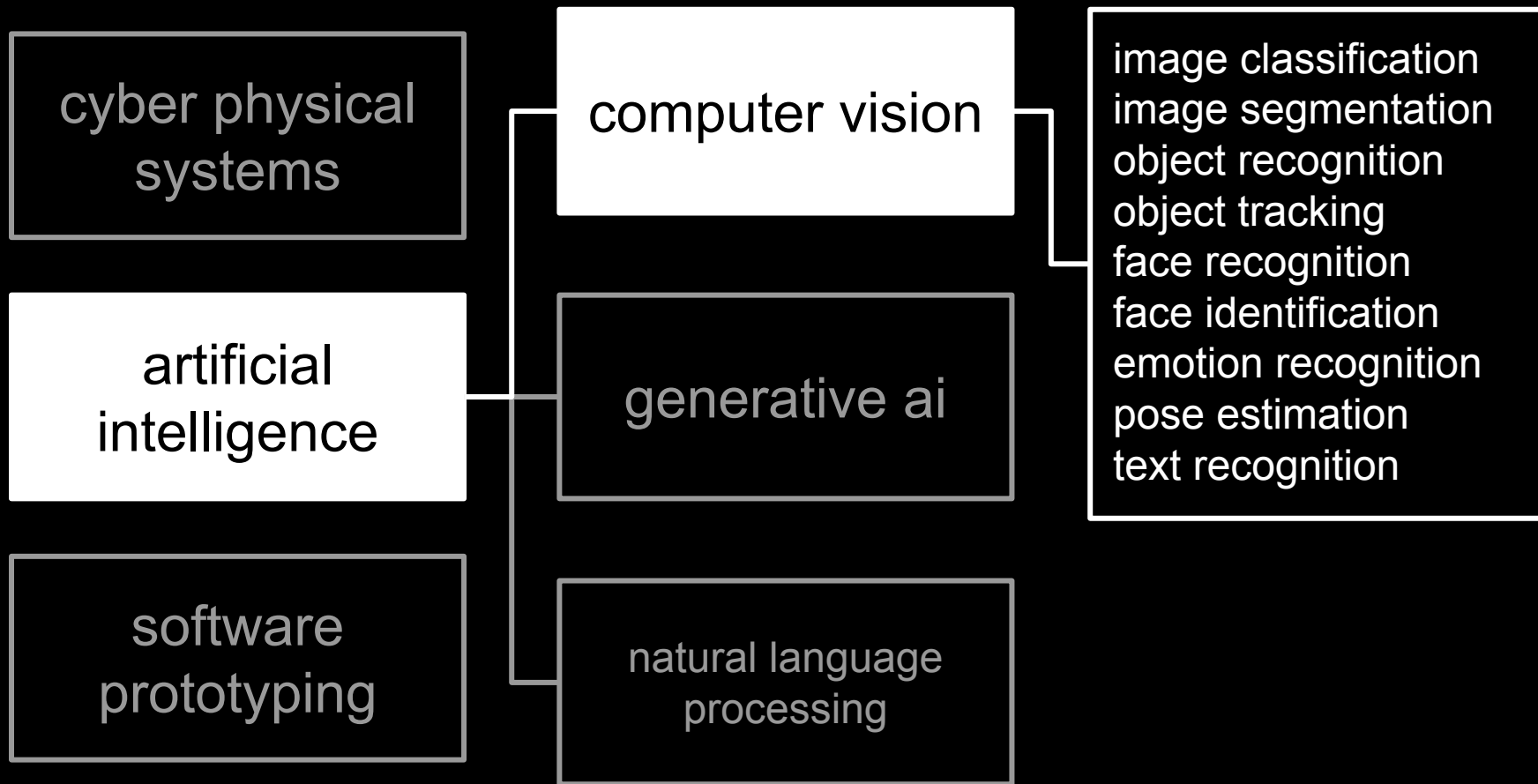
cyber physical
systems

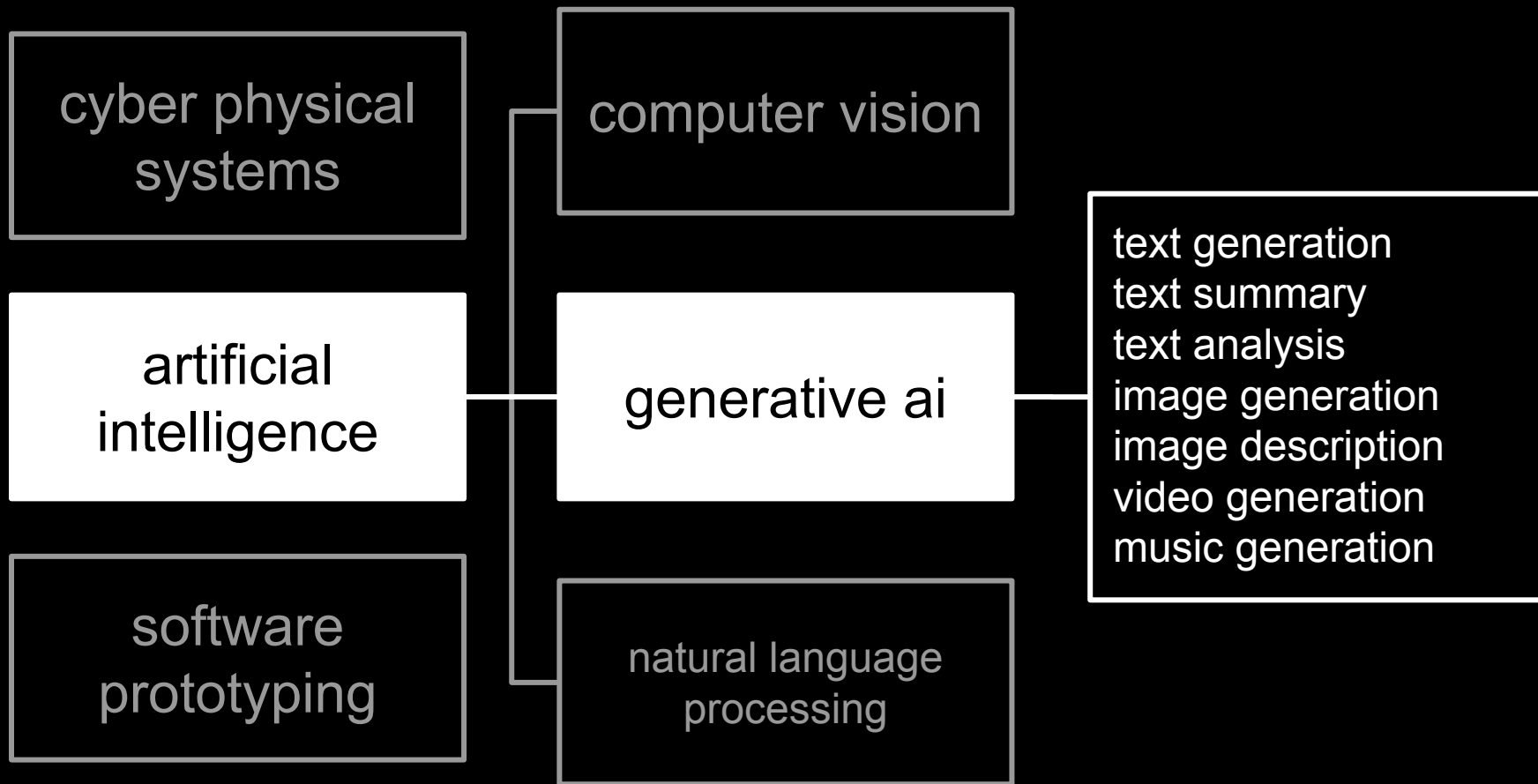
artificial
intelligence

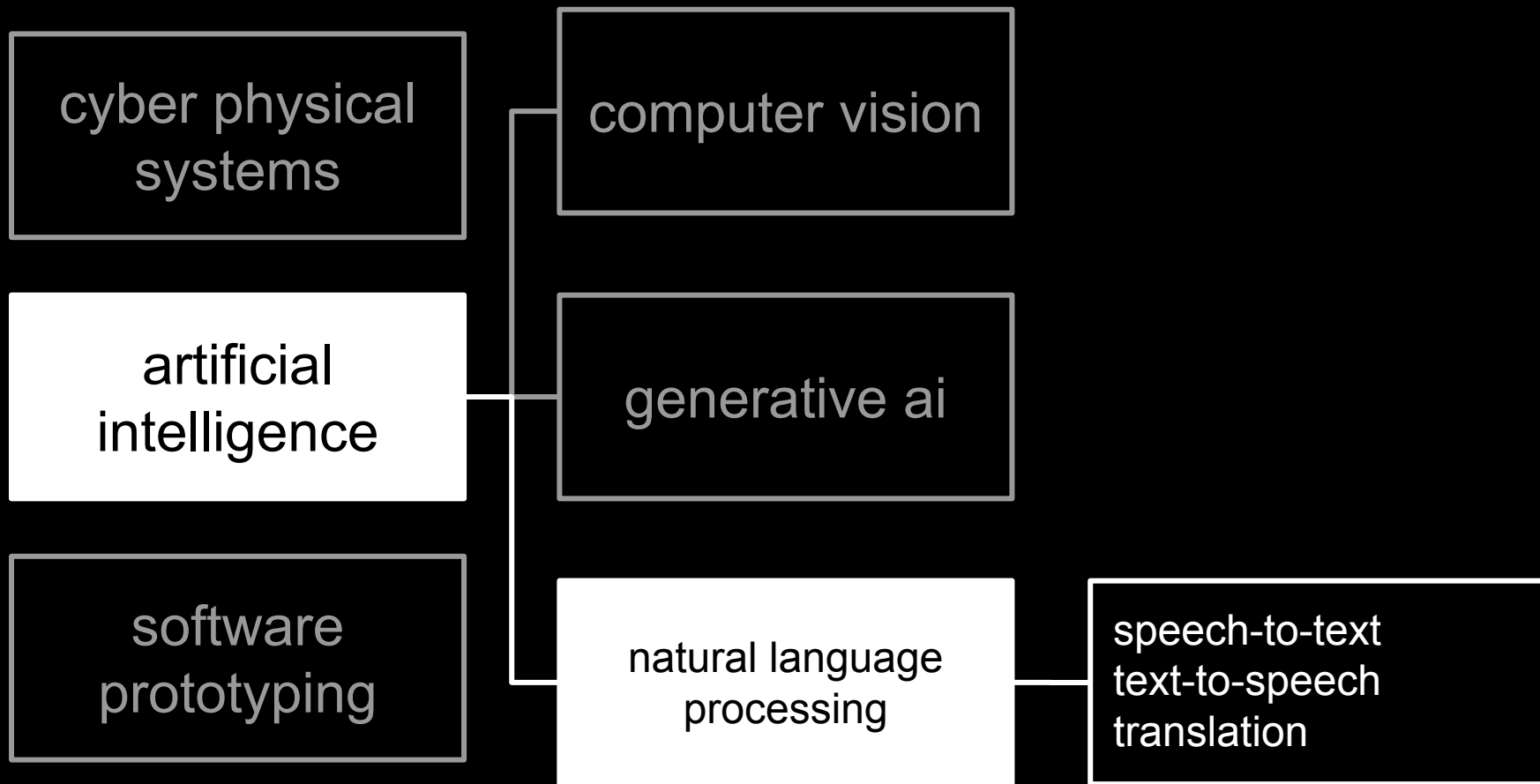
software
prototyping

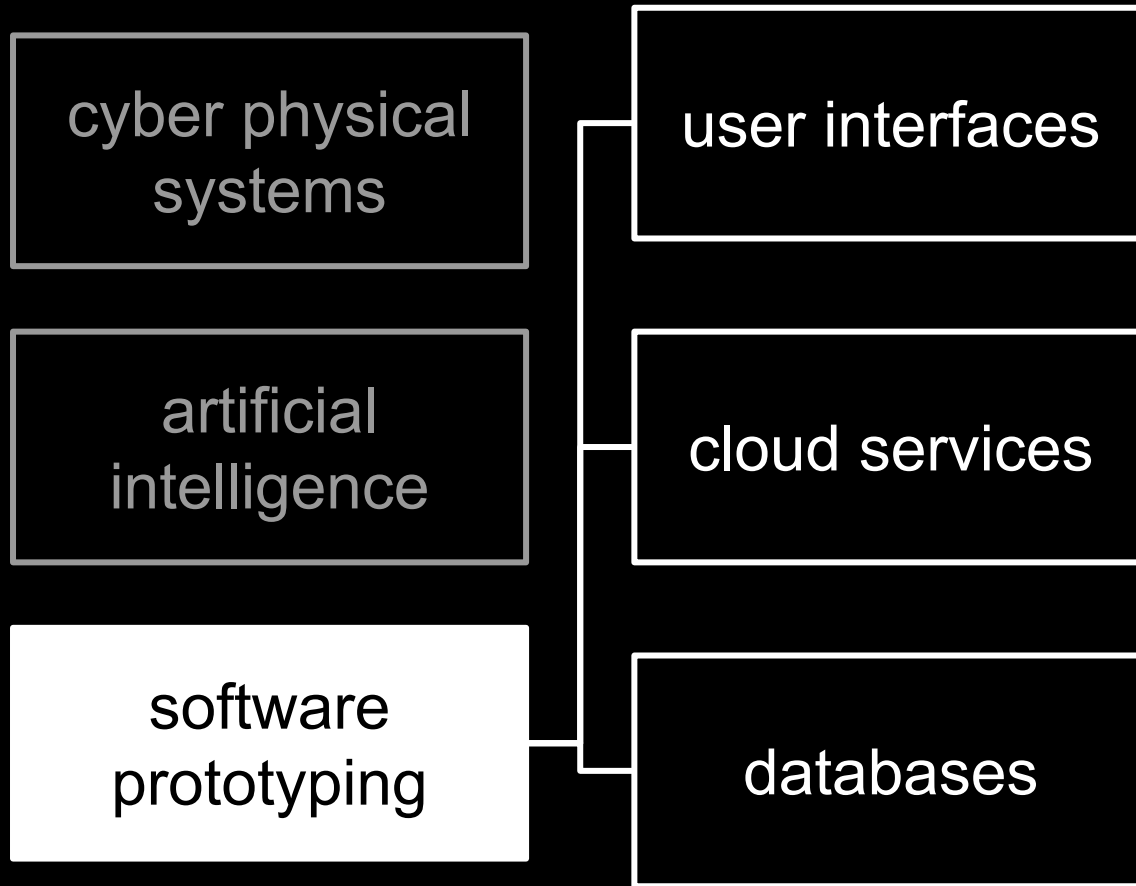












introductory example

visual studio code
programs
python

LEDs

large language models

speech-to-text

user interface

SENSORS

temperature / humidity

rgb led button

camera

thermal imaging camera

microphone

keyboard

temperature / humidity

```
th = BrickletHumidityV2(UID, ipcon)...
```

```
th.get_humidity()
```

```
th.get_temperature()
```

```
th.register_callback(th.CALLBACK_HUMIDITY, cb_humidity)  
th.register_callback(th.CALLBACK_TEMPERATURE, ...)
```

```
th.set_humidity_callback_configuration(250, False, "x", 0, 0)  
th.set_temperature_callback_configuration(...)
```

rgb led button


```
btn = BrickletRGBLEDButton(UID, ipcon)...
```

```
btn.set_color(255, 0, 0)
```

```
btn.get_button_state()
```

```
btn.register_callback(...)
```

camera

OpenCV

```
import cv2
```



```
# Get video capture device (webcam)  
webcam = cv2.VideoCapture(0)
```



```
# Read a frame
```

```
success, frame = webcam.read()
```




```
# Show the image from the frame  
cv2.imshow("Webcam", frame)
```



```
# Save the frame as .png
```

```
cv2.imwrite("screenshot.png", frame)
```

thermal imaging camera

OpenCV

Tinkerforge



```
ti = BrickletThermalImaging(UID, ipcon)
ti.set_image_transfer_config(...)
img = ti.get_high_contrast_image()
```

```
ti.register_callback(...)
```

microphone

```
import pyaudio
```



```
# Define recording parameters  
FORMAT = pyaudio.paInt16  
CHANNELS = 1  
RATE = 44100  
CHUNK = 1024
```

```
# Get access to the microphone  
audio = pyaudio.PyAudio()
```

```
# Start listening  
stream = audio.open(...)
```

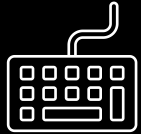
```
# Read a chunk of frames  
stream.read(CHUNK)
```

```
# Stop and close stream  
stream.stop_stream()  
stream.close()
```

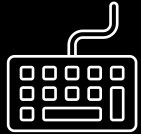
```
# Terminate access to microphone  
audio.terminate()
```

keyboard

```
import keyboard
```

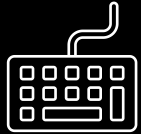



```
# Define a callback function for a key  
def record_audio():  
    print("Recording audio...")
```



```
# Add key listener
```

```
keyboard.add_hotkey("r", record_audio)
```



```
# Wait until a specific key was pressed  
keyboard.wait("esc")
```

ACTUATORS

rgb led
OLED display
speaker

rgb led

```
led = BrickletRGBLEDV2(UID, ipcon)  
oled.set_rgb_value(255, 0, 0)
```

OLED display


```
oled = BrickletOLED128x64V2(UID, ipcon)
oled.clear_display()
oled.write_line(0, 0, "Welcome!")
```

speaker

```
import simpleaudio as sa
```



```
# Create a wave object from .wav-file and play it  
wav = sa.WaveObject.from_wave_file("sound.wav")  
wav.play().wait_done()
```