Controlling lab and production equipment with Python

Pycon Sweden 2024-11-15
by Ulrik Södergren
ulrik@oddtech.se

Controlling lab and production equipment with Python

Pycon Sweden 2024-11-15
by Ulrik Södergren
ulrik@oddtech.se

About Ulrik

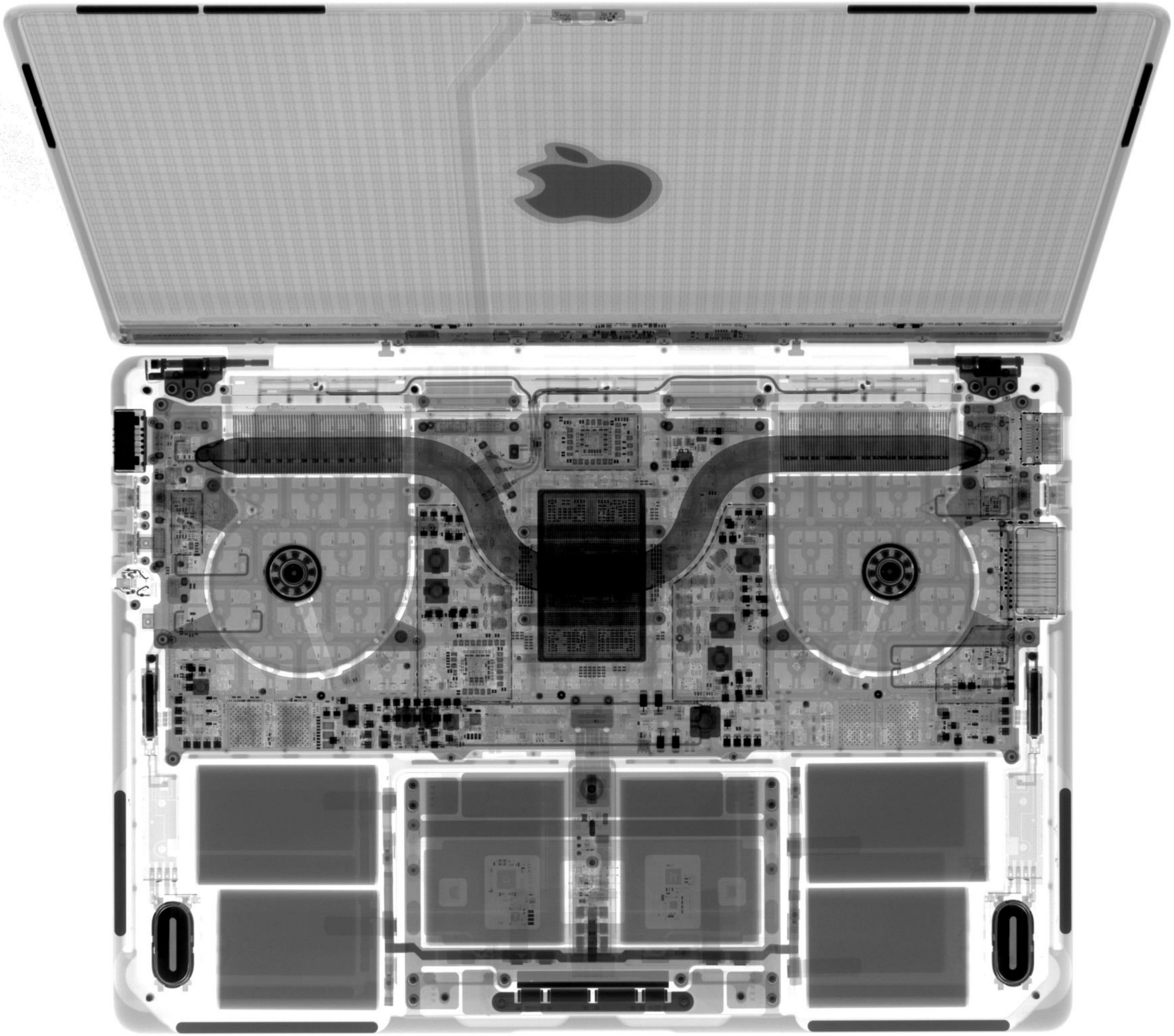
- Ulrik Södergren
- ulrik@oddtech.se
- OddTech / DigitalFotografen
- Clients: IKEA, Hasselblad, Texdot, Folksam, Öresundsbron, Luxbright
- Used to write for DatorMagazin

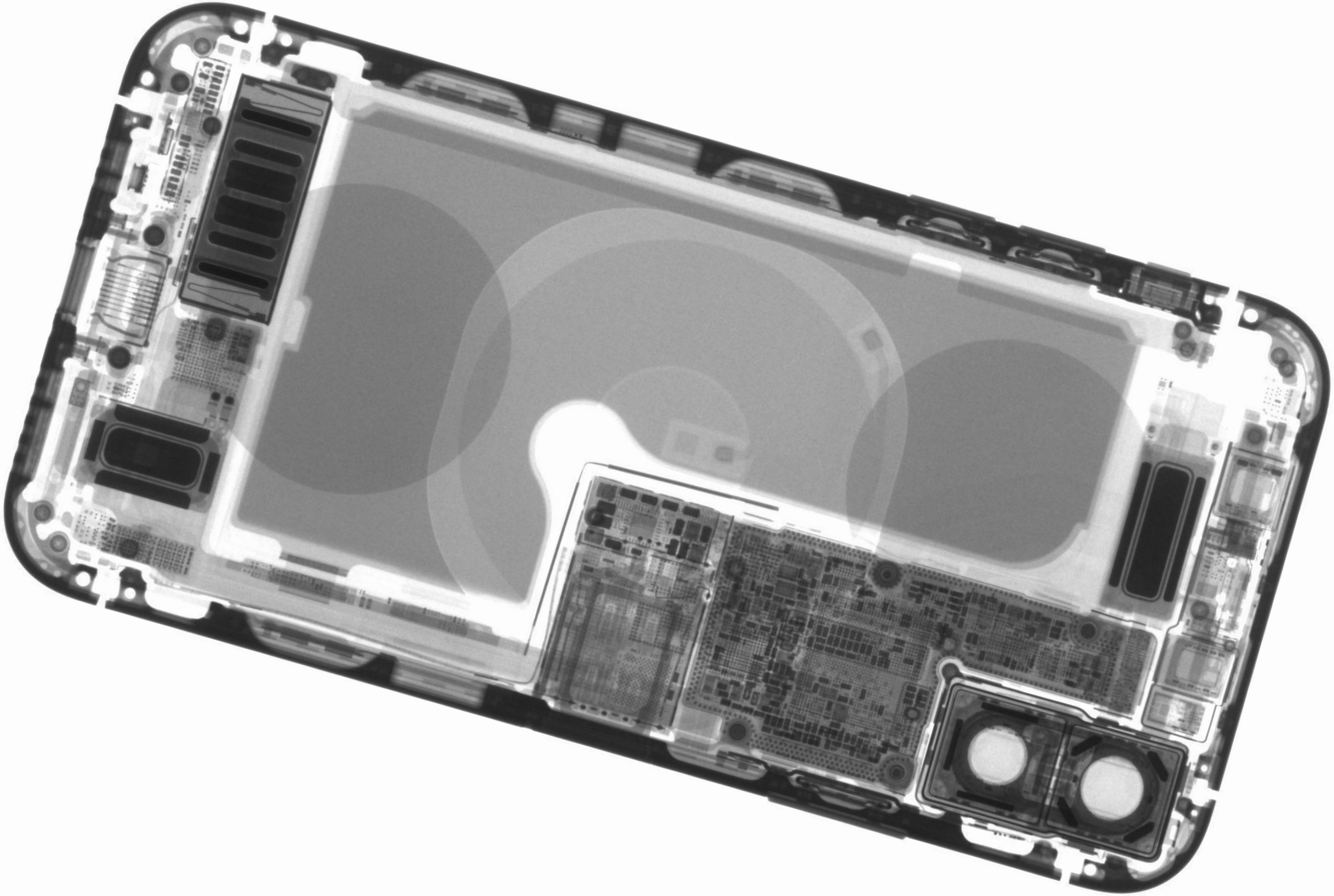
ODDTECH

About Luxbright

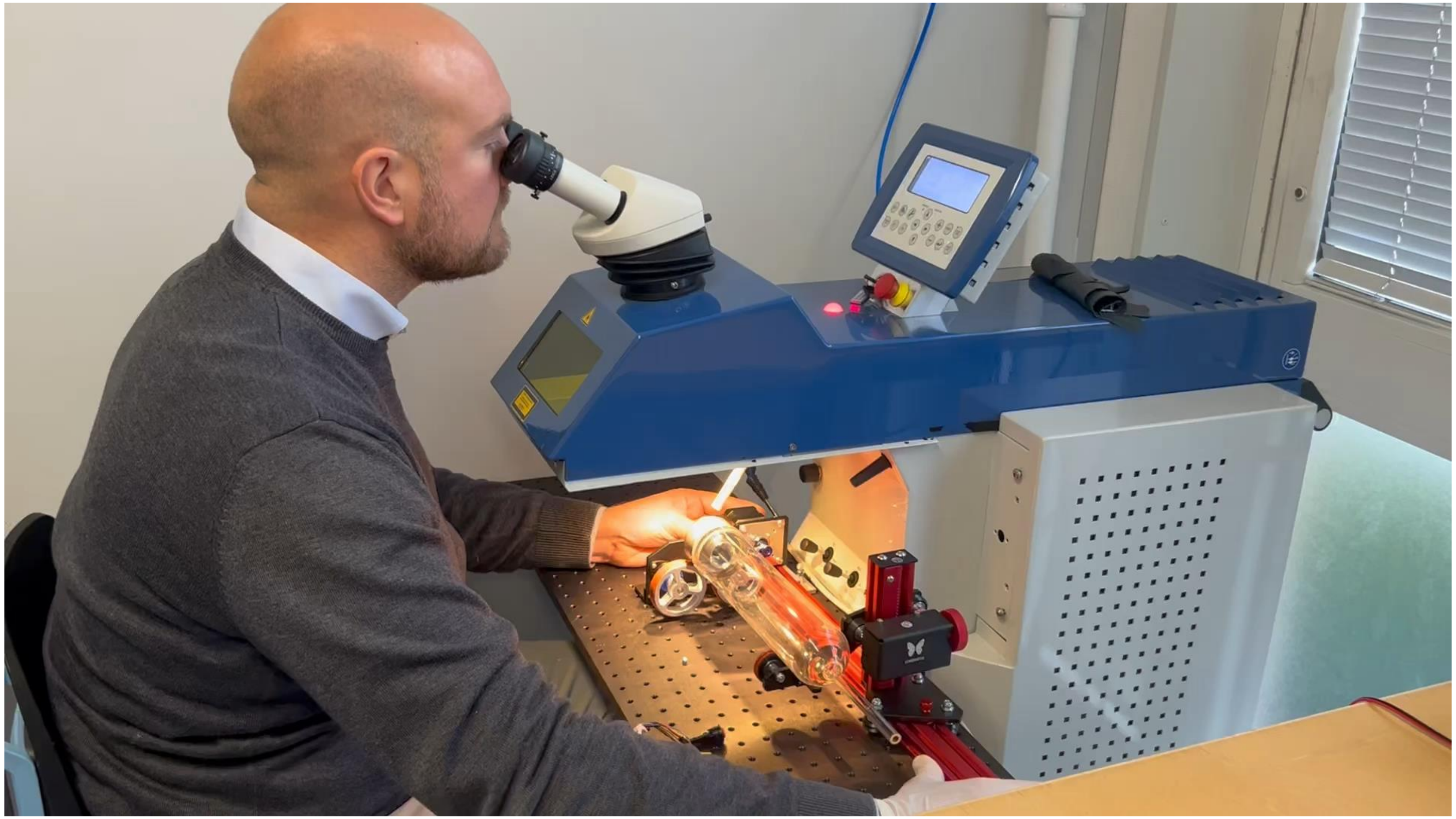
- Develops and manufactures X-ray tubes in Göteborg, Sweden
- Production moved from China to Sweden 2022
- Microfocus X-ray tubes
- Cold cathode X-ray tubes
- Custom made X-ray tubes

LUXBRIGHT

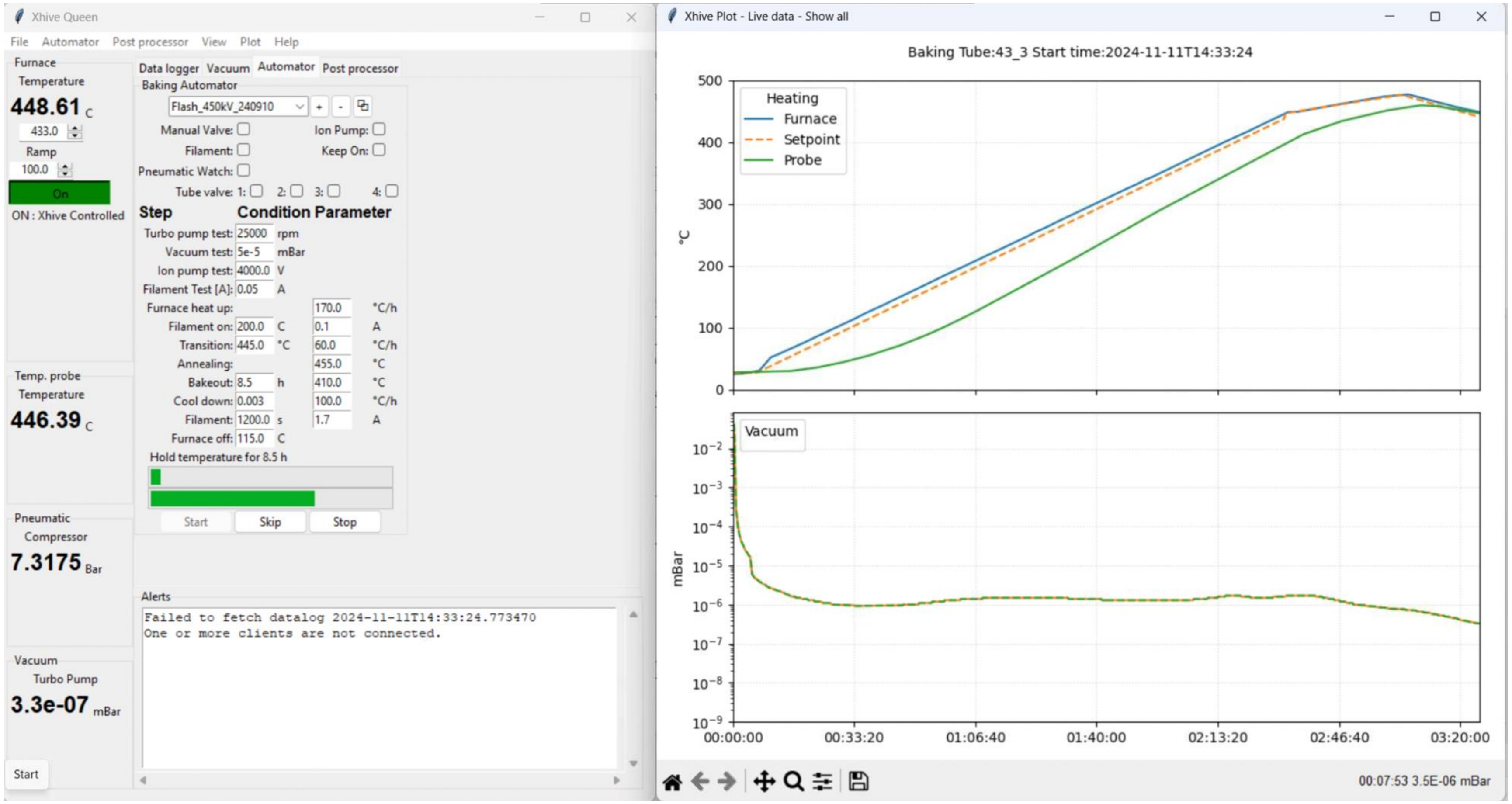




Laser welder and rotating fixture controlled by Python application on Raspberry Pi



Control and data logging framework



Xhive Queen

Furnace Temperature: 448.61 C

433.0

Ramp: 100.0

On

ON : Xhive Controlled

Baking Automator: Flash_450kV_240910

Manual Valve: □ Ion Pump: □

Filament: □ Keep On: □

Pneumatic Watch: □

Tube valve: 1: □ 2: □ 3: □ 4: □

Step	Condition		Parameter	
Turbo pump test:	25000	rpm		
Vacuum test:	5e-5	mBar		
Ion pump test:	4000.0	V		
Filament Test [A]:	0.05	A		
Furnace heat up:			170.0	°C/h
Filament on:	200.0	C	0.1	A
Transition:	445.0	°C	60.0	°C/h
Annealing:			455.0	°C
Bakeout:	8.5	h	410.0	°C
Cool down:	0.003		100.0	°C/h
Filament:	1200.0	s	1.7	A
Furnace off:	115.0	C		

Hold temperature for 8.5 h

Temp. probe Temperature: 446.39 C

Pneumatic Compressor: 7.3175 Bar

Vacuum Turbo Pump: 3.3e-07 mBar

Alerts

```
Failed to fetch datalog 2024-11-11T14:33:24.773470
One or more clients are not connected.
```

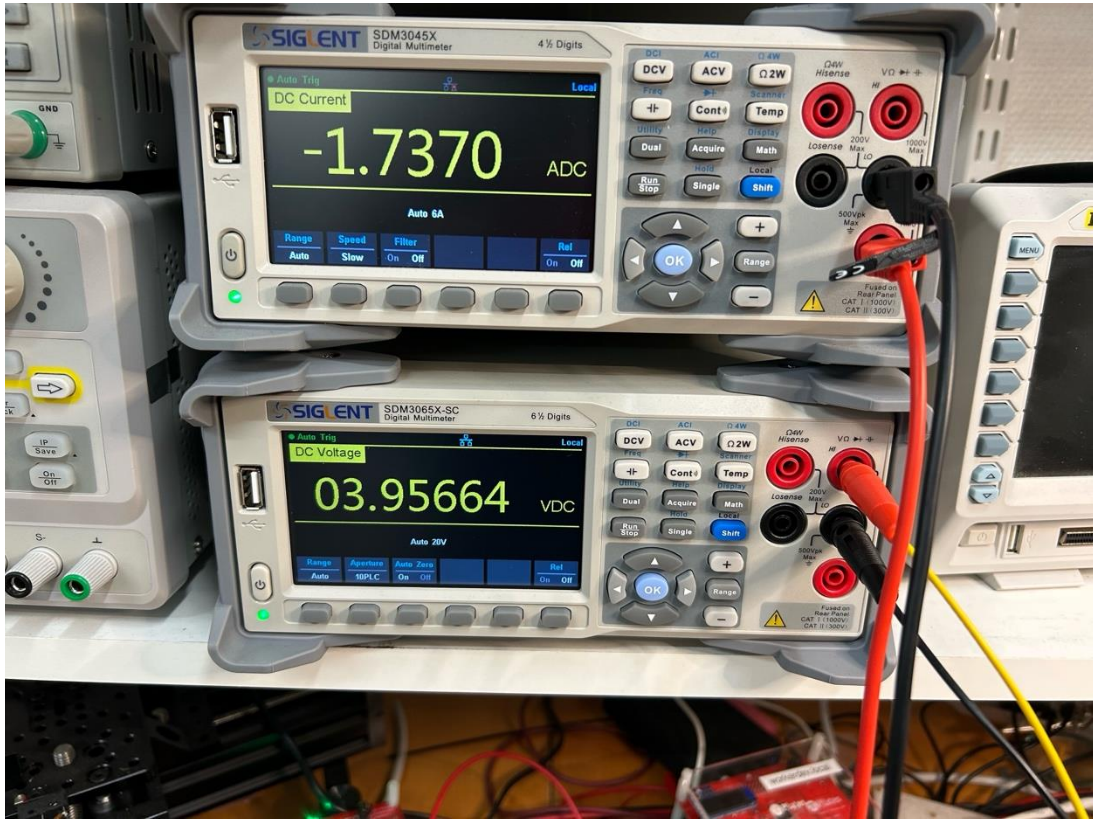
Xhive Plot - Live data - Show all

Baking Tube:43_3 Start time:2024-11-11T14:33:24

00:07:53 3.5E-06 mBar

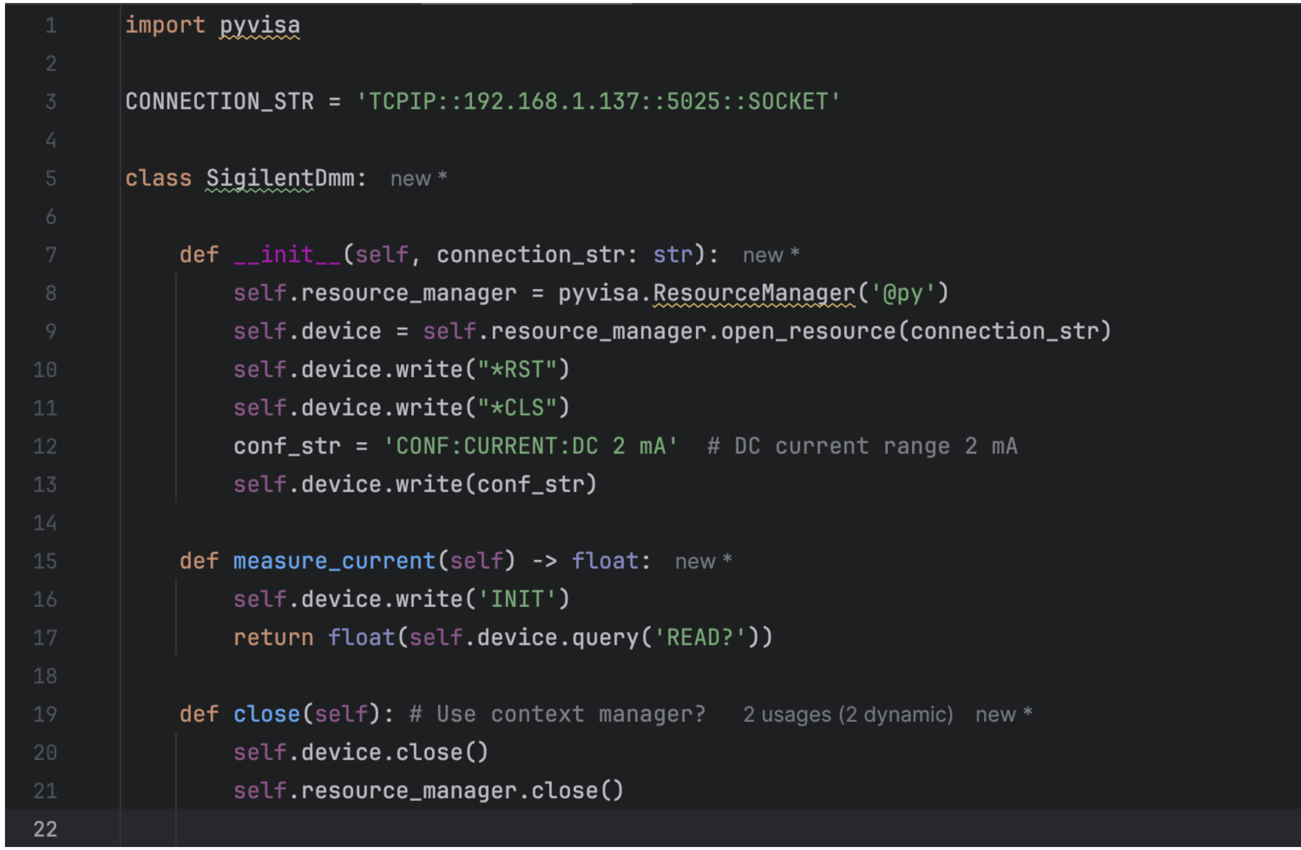
Automating lab equipment

- A common solution is to use LabView
- Power supplies, oscilloscopes, multimeters, signal generators…
- Virtual Instrument Software Architecture (VISA)



Control instruments with PyVISA

- Virtual Instrument Software Architecture (VISA)
- The command language is called SCPI
- GPIB, RS232, USB or Ethernet (VXI, PXI, LXI)
- Integrates well with Pytest

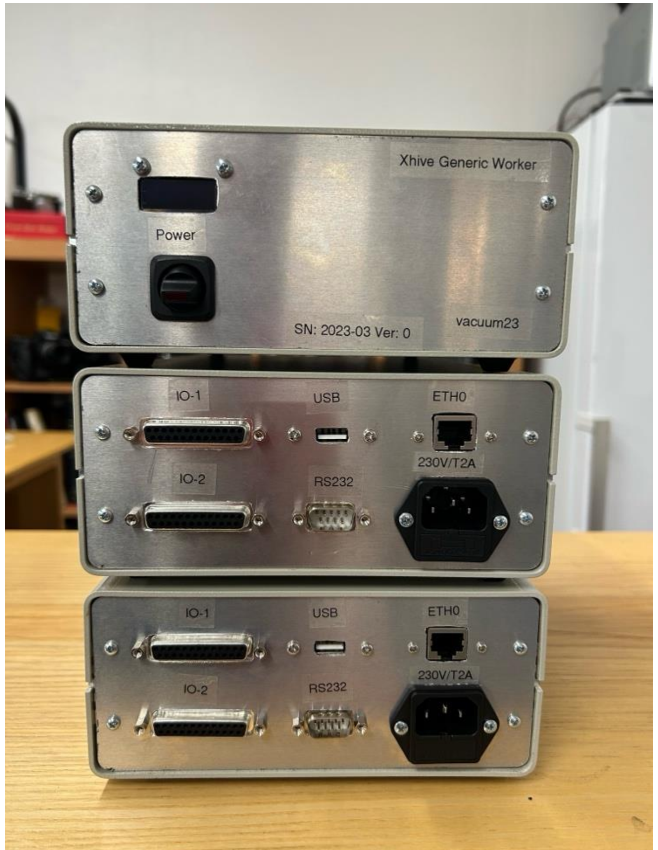


```python
import pyvisa

CONNECTION_STR = 'TCPIP::192.168.1.137::5025::SOCKET'

class SigilentDmm:

    def __init__(self, connection_str: str):
        self.resource_manager = pyvisa.ResourceManager('@py')
        self.device = self.resource_manager.open_resource(connection_str)
        self.device.write("*RST")
        self.device.write("*CLS")
        conf_str = 'CONF:CURRENT:DC 2 mA'  # DC current range 2 mA
        self.device.write(conf_str)

    def measure_current(self) -> float:
        self.device.write('INIT')
        return float(self.device.query('READ?'))

    def close(self): # Use context manager?
        self.device.close()
        self.resource_manager.close()

```

Automating (small) industrial processes

- A common solution is PLC (Programmable Logic Controller)

- But why not use Raspberry Pi:s and Python?



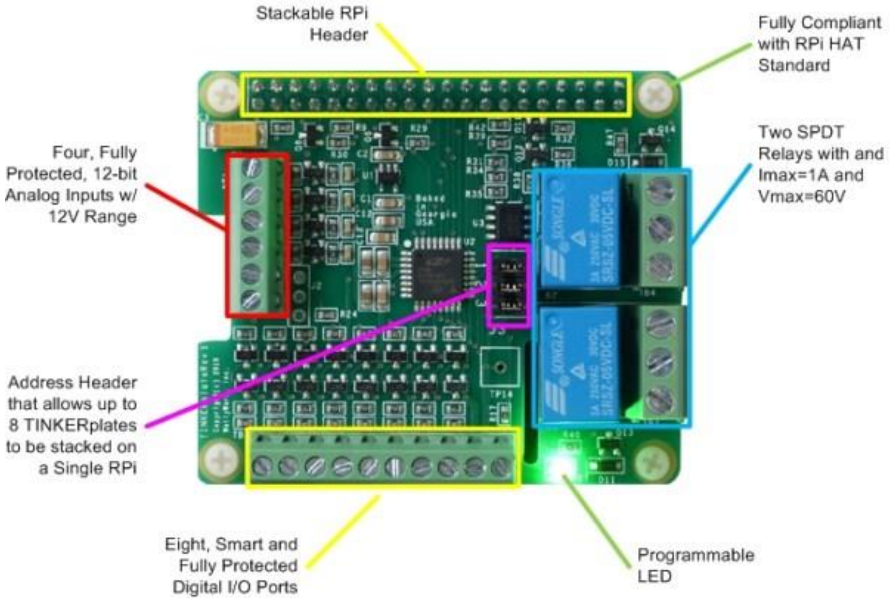
Analogue industrial sensors

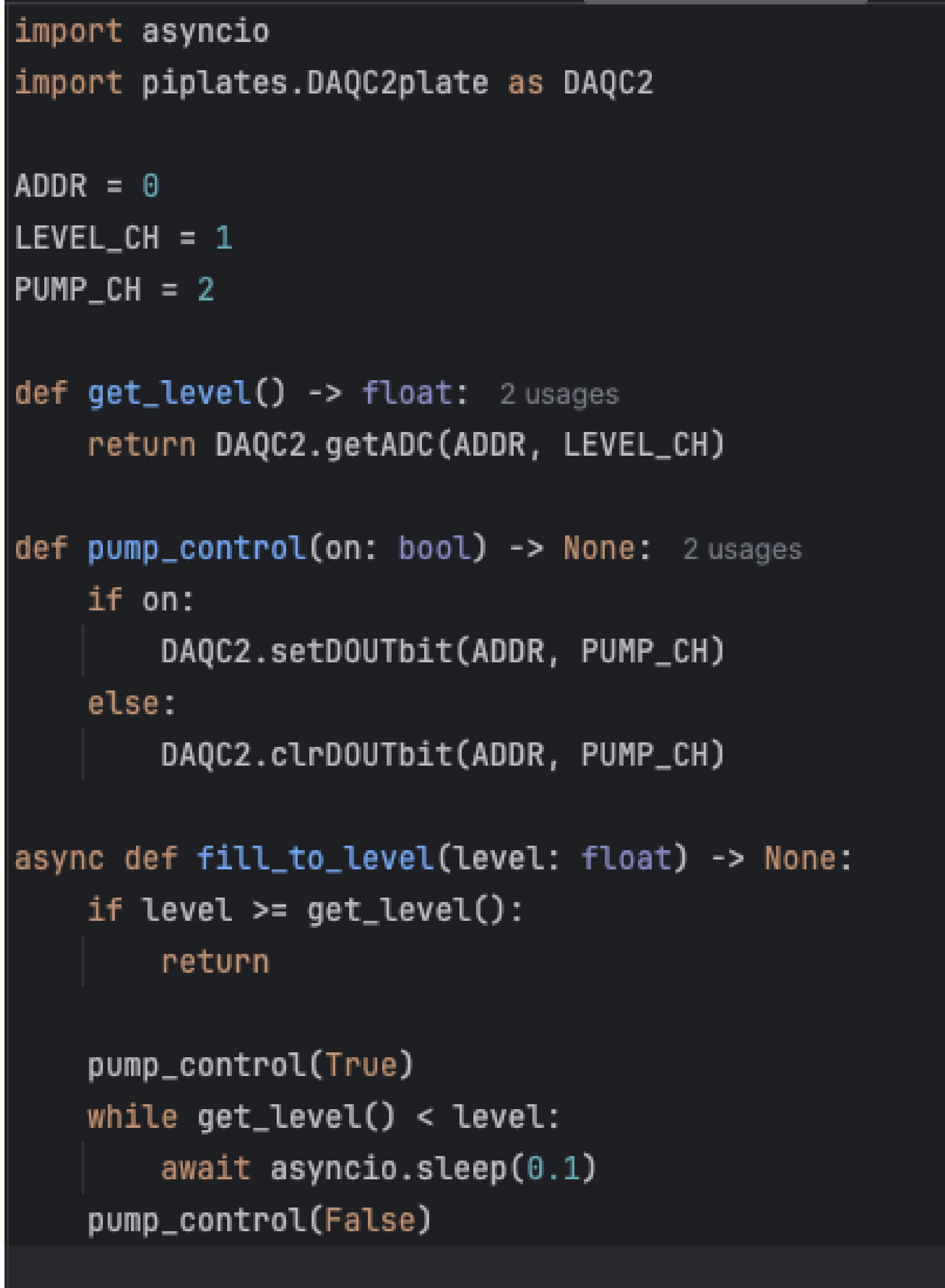
- On/off
- Pulses
- 0-10V
- 4-20 mA

- Proximity
- Level or position
- Flow or speed
- Pressure
- Voltage

Raspberry Pi + DAQ Card

- Data Acquisition Card (DAQC)
- Multiple analogue and digital inputs
- Pi-Plates, MCC

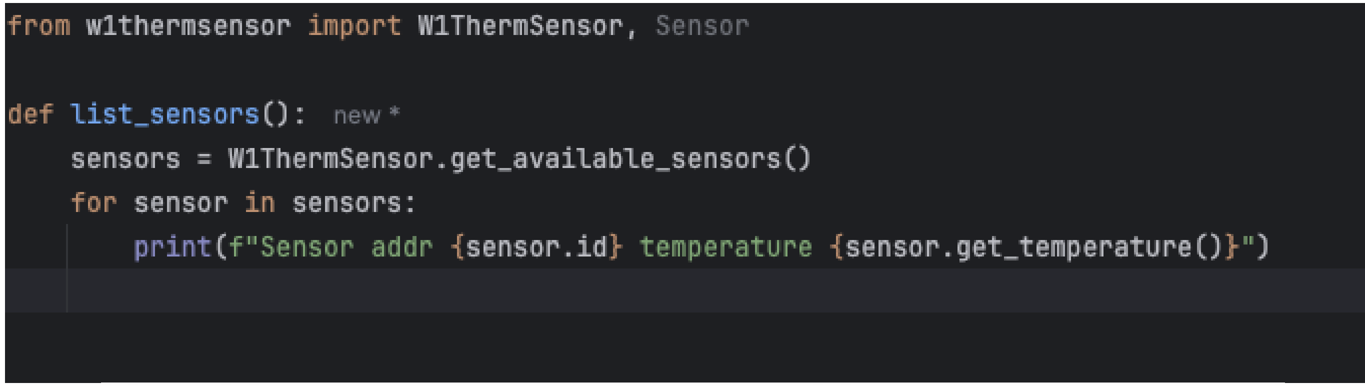

Stackable RPi Header
Fully Compliant with RPi HAT Standard
Four, Fully Protected, 12-bit Analog Inputs w/ 12V Range
Two SPDT Relays with and Imax=1A and Vmax=60V
Address Header that allows up to 8 TINKERplates to be stacked on a Single RPi
Eight, Smart and Fully Protected Digital I/O Ports
Programmable LED



```python
import asyncio
import piplates.DAQC2plate as DAQC2

ADDR = 0
LEVEL_CH = 1
PUMP_CH = 2

def get_level() -> float:  2 usages
    return DAQC2.getADC(ADDR, LEVEL_CH)

def pump_control(on: bool) -> None:  2 usages
    if on:
        DAQC2.setDOUTbit(ADDR, PUMP_CH)
    else:
        DAQC2.clrDOUTbit(ADDR, PUMP_CH)

async def fill_to_level(level: float) -> None:
    if level >= get_level():
        return

    pump_control(True)
    while get_level() < level:
        await asyncio.sleep(0.1)
    pump_control(False)
```

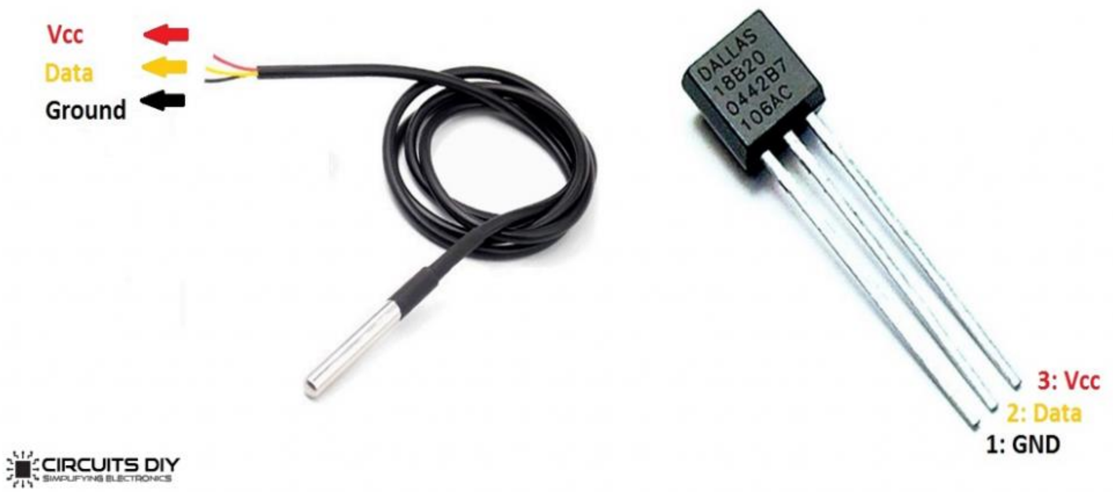
1-wire thermometers

- DS18B20 (-55 to +125°C)
- Can have multiple sensors in parallel on same cable
- Each sensor has a globally unique address



```
from w1thermsensor import W1ThermSensor, Sensor

def list_sensors():  new *
    sensors = W1ThermSensor.get_available_sensors()
    for sensor in sensors:
        print(f"Sensor addr {sensor.id} temperature {sensor.get_temperature()}")
```

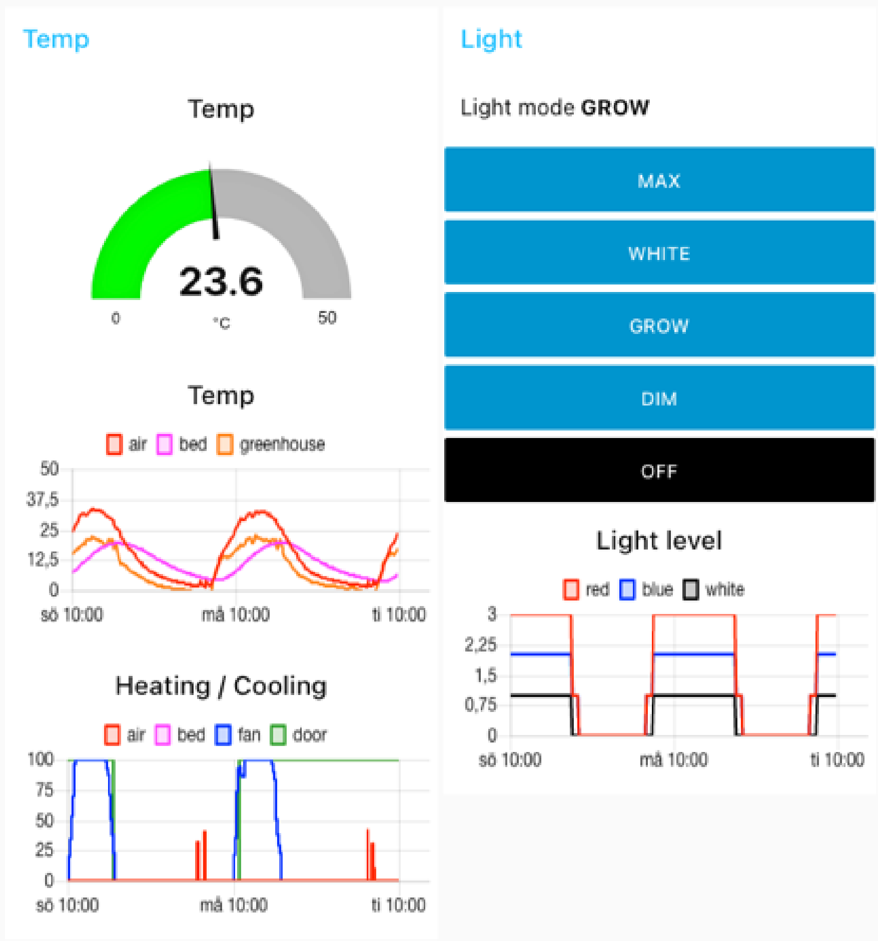
DS18B20 Temperature Sensor Pinout



CIRCUITS DIY
SIMPLIFYING ELECTRONICS

Greenhouse or home brewery

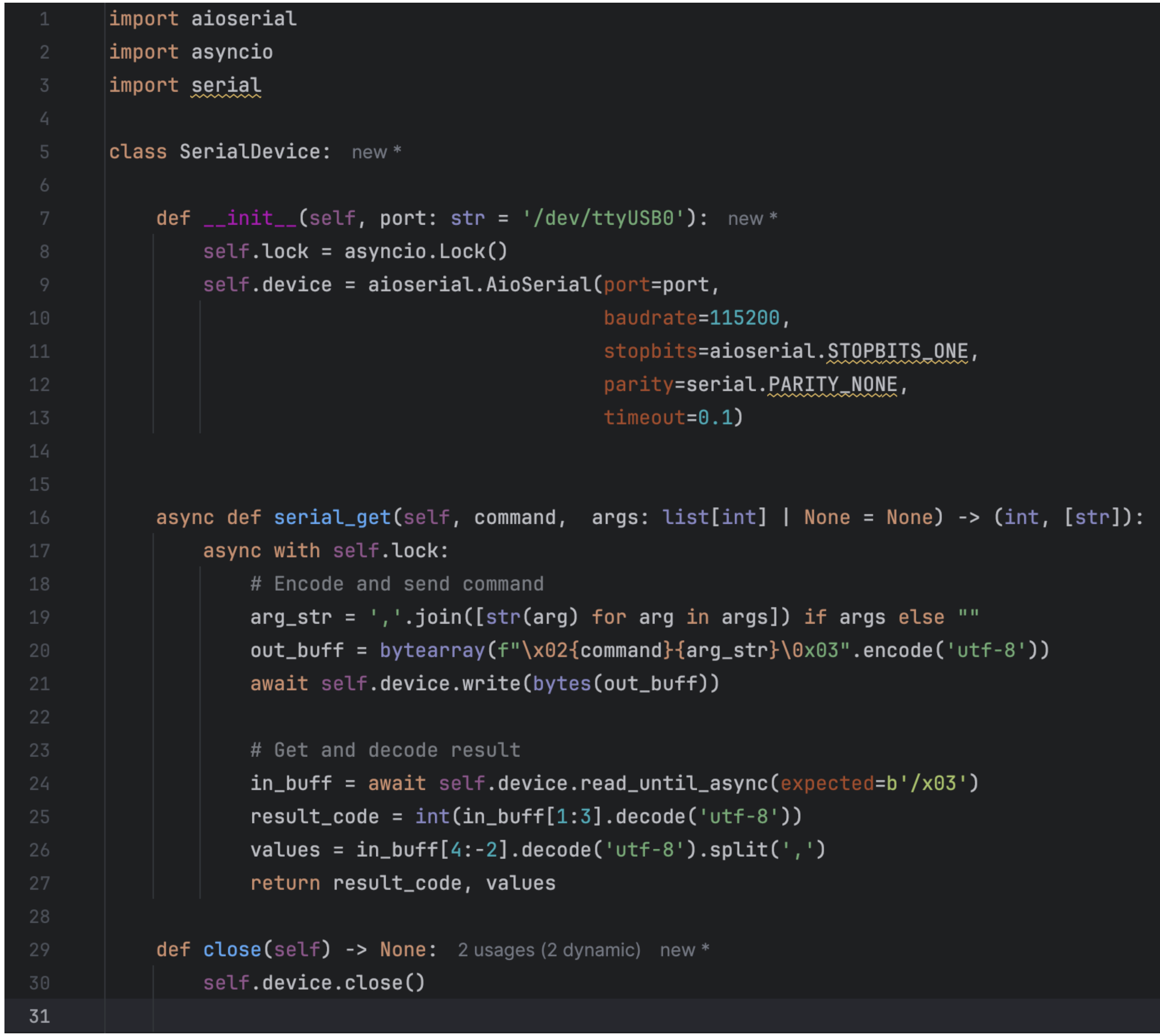
- Measure temperature via 1-wire
- Control and monitor other properties via DAQ Card
- Control heaters via relays



Light mode **GROW**

Serial ports

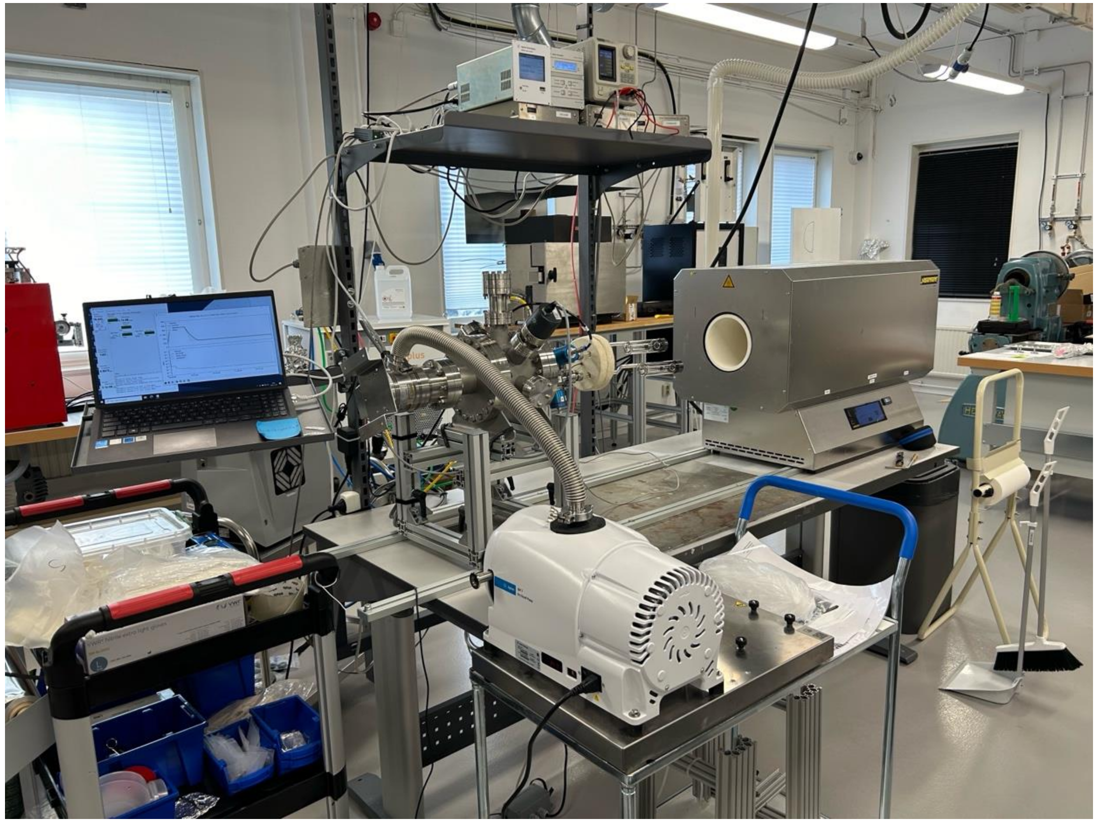
- RS232, RS485, RS422, USB, Ethernet
- Use aioserial to get async support
- ModBus – Use PyModbus to communicate with PLC:s



```
import aioserial
import asyncio
import serial

class SerialDevice:  new *

    def __init__(self, port: str = '/dev/ttyUSB0'):  new *
        self.lock = asyncio.Lock()
        self.device = aioserial.AioSerial(port=port,
                                          baudrate=115200,
                                          stopbits=aioserial.STOPBITS_ONE,
                                          parity=serial.PARITY_NONE,
                                          timeout=0.1)


    async def serial_get(self, command,  args: list[int] | None = None) -> (int, [str]):
        async with self.lock:
            # Encode and send command
            arg_str = ','.join([str(arg) for arg in args]) if args else ""
            out_buff = bytearray(f"\x02{command}{arg_str}\0x03".encode('utf-8'))
            await self.device.write(bytes(out_buff))

            # Get and decode result
            in_buff = await self.device.read_until_async(expected=b'/x03')
            result_code = int(in_buff[1:3].decode('utf-8'))
            values = in_buff[4:-2].decode('utf-8').split(',')
            return result_code, values

    def close(self) -> None:  2 usages (2 dynamic)  new *
        self.device.close()

```

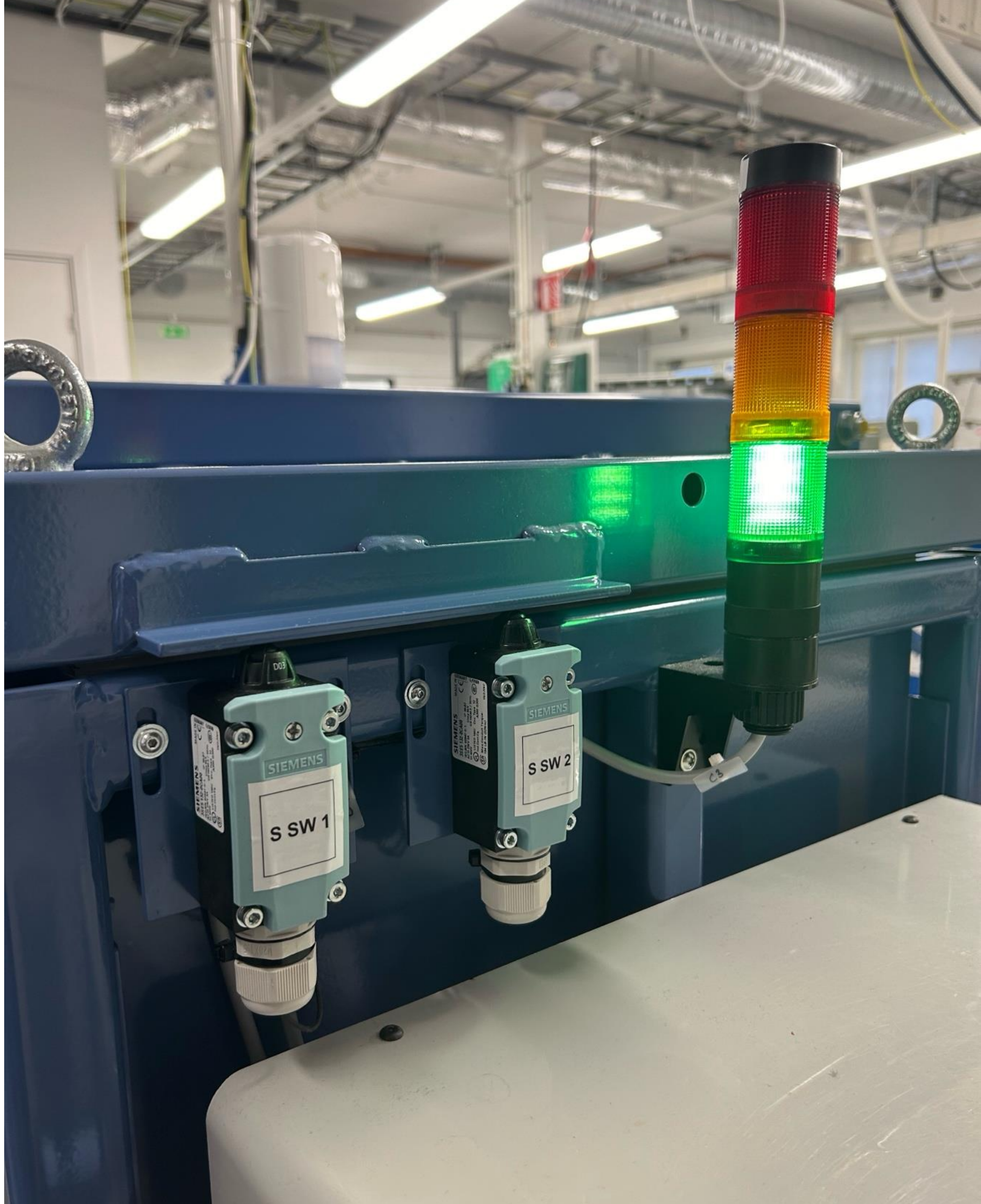
Challenges when testing software that drive physical hardware

- Don’t expect exact values
- Timing problems
- Slow tests
- The hardware is not always available – create simulators
- Bugs might break expensive hardware



Safety first

- Safety breakers
- Warning lights
- Use interlocks on doors
- Use multiple layers. Implement safety systems in both hardware and software
- And remember Murphys Law

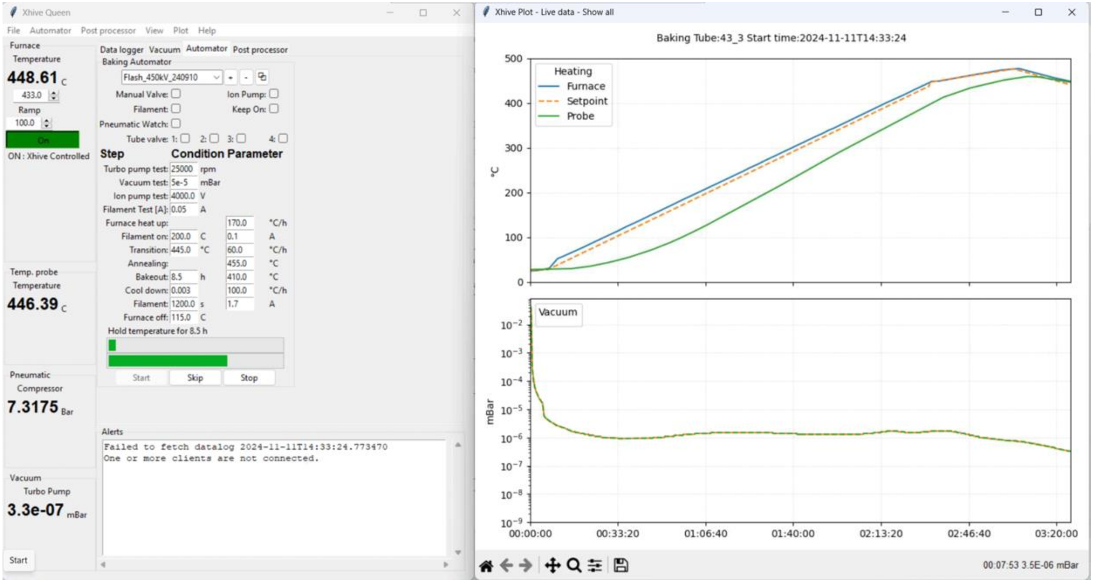


Our Python automation journey

- Analysis of X-ray test images
- Control of High Voltage supply in electron emission lab
- Automating the X-ray research lab
- Automation of production processes
- Distributed system of Raspberry Pi:s + Windows GUI
- Internally developed Python data logging and control framework

Xhive framework

- Distributed modular control and data logging using PC/Mac GUI app + several Raspberry Pi nodes
- Manual or fully automated operation
- Logs are stored locally and merged at end of session
- 0MQ communication protocol
- For internal use only,
  but there are plans to release framework as Open Source



Furnace
Temperature
448.61 C
433.0
Ramp
100.0
On
ON : Xhive Controlled

Temp. probe
Temperature
446.39 C

Pneumatic
Compressor
7.3175 Bar

Vacuum
Turbo Pump
3.3e-07 mBar

Baking Automator
Flash_450kV_240910

Manual Valve: □ Ion Pump: □
Filament: □ Keep On: □
Pneumatic Watch: □
Tube valve: 1: □ 2: □ 3: □ 4: □

Step	Condition		Parameter	
Turbo pump test:	25000	rpm		
Vacuum test:	5e-5	mBar		
Ion pump test:	4000.0	V		
Filament Test [A]:	0.05	A		
Furnace heat up:			170.0	°C/h
Filament on:	200.0	C	0.1	A
Transition:	445.0	°C	60.0	°C/h
Annealing:			455.0	°C
Bakeout:	8.5	h	410.0	°C
Cool down:	0.003		100.0	°C/h
Filament:	1200.0	s	1.7	A
Furnace off:	115.0	C		

Hold temperature for 8.5 h

Alerts

```
Failed to fetch datalog 2024-11-11T14:33:24.773470
One or more clients are not connected.
```

Baking Tube:43_3 Start time:2024-11-11T14:33:24

00:07:53 3.5E-06 mBar

Some links

- Raspberry Pi https://www.raspberrypi.com/
- Pi-Plates: https://pi-plates.com/
- Electrokit: https://www.electrokit.com
- Kjell & Co: https://www.kjell.com/
- Pyvisa: https://pyvisa.readthedocs.io
- PyModbus: https://pymodbus.readthedocs.io
- Ask Ulrik: ulrik@oddtech.se

Thanks for listening

ulrik@oddtech.se