

TEMPERA

Zeitaufzeichnung und Raumklima

G4T1

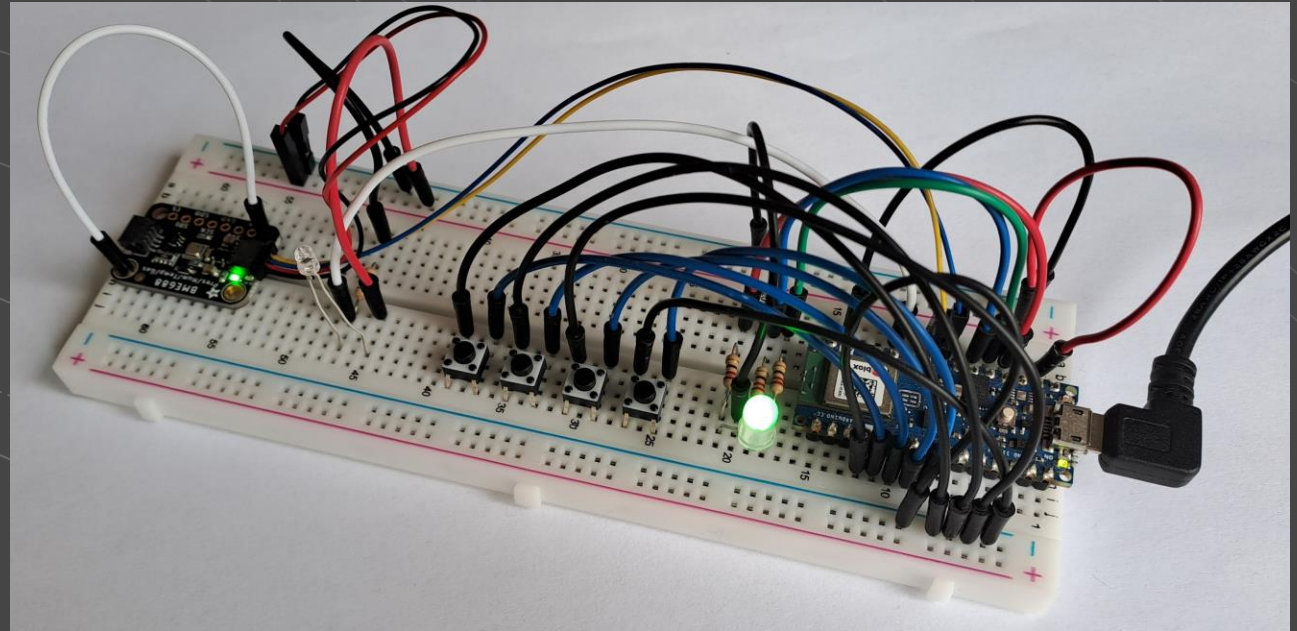
```
// Set LED-color for different work modes (r-g-b)
#define DW_COLOR {0, 0, 255}
#define MT_COLOR {255, 40, 10}
#define OO_COLOR {255, 0, 0}
#define PT_COLOR {0, 64, 0}

// Delay in ms after which a new button press will
#define BUTTON_COOLDOWN 600

// Update interval in ms after which the station
#define UPDATE_INTERVAL_TIME 60000

// Update interval in ms after which the station
#define UPDATE_INTERVAL_RC 60000

// Device name and custom id
#define DEVICE_NAME "G4T1-Tempera-Station #1"
#define DEVICE_SN "tempera_station_1"
```



```
// ##### BLE SETUP #####

// Set up the device information service
BLEService deviceInformationService("180A");
BLEStringCharacteristic manufacturerNameCharacteristic("2A29", BLERead, 64);
BLEStringCharacteristic serialNumberCharacteristic("2A25", BLERead, 64);

// Set up the elapsed time service for time tracking
BLEService elapsedTimeService("183F");
BLECharacteristic currentElapsedTimeCharacteristic("2BF2", BLERead | BLEIndicate, sizeof(elapsedTimeCharacteristicStructure));


// Set up the environmental sensing service for room climate measurements
BLEService environmentalSensingService("181A");
BLECharacteristic temperatureCharacteristic("2A6E", BLERead, sizeof(roomClimateData.temperature));
BLECharacteristic irradianceCharacteristic("2A77", BLERead, sizeof(roomClimateData.irradiance));
BLECharacteristic humidityCharacteristic("2A6F", BLERead, sizeof(roomClimateData.humidity));
BLECharacteristic nmvocCharacteristic("2BD3", BLERead, sizeof(roomClimateData.nmvoc));
```

```

async def main():
    try:
        async with asyncio.TaskGroup() as tg:
            _ = tg.create_task(utils.init_globals())
            tempera_station = tg.create_task(bleclient.discovery_loop())
        # Use except* to catch a group (list) of the same exceptions. Must be used with
        # the async task manager as multiple exceptions can be thrown because of the
        # A simple catch wouldn't allow them all to be handled.
    except* BluetoothOffException:
        logger.critical("Bluetooth is off. Turn on Bluetooth and try again :)")
        sys.exit(0)
    except* RuntimeError:
        logger.info("Returning to device discovery.")
        raise

```

compose.yaml x



```

46 >> services:
45 >   ble-app:
44     build: ./tempera-accesspoint
43     volumes:
42       - /var/run/dbus:/var/run/dbus
41         # shared volume between host and docker, so that
40         # the host machine, and you don't have to re-
39       - ./tempera-accesspoint:/home
38
37 >   back-end:
36     build:

```

`async` `tempera.bleclient.etl.measurements_handler(client: BleakClient, characteristics: List[BleakGATTCharacteristic])`

Read the measurement service characteristics from the device and create/save the resulting measurement to db.

- Parameters:**
- **client** – the connection to the tempera station.
 - **characteristics** – list of measurement characteristics to read from the tempera station.
- Returns:** None
- Raises:**
- **bleak.exc.BleakError** – if connection issues occur when reading the characteristics from the tempera station.
 - **bleak.exc.BleakDBusError** – see BleakError

🏠 / BLE app

[View page source](#)

BLE app

Configuration script

A python configuration script `configure.py` is provided to configure the parameters necessary for running the BLE application.

⚠ Warning

Setting the parameters by manually writing/changing the `conf.yaml` parameter file can easily cause problems or program malfunction. Use the provided script.

compose.yaml

```
46 services:
45   ble-app:
44     build: ./tempera-accesspoint
43     volumes:
42       - /var/run/dbus:/var/run/dbus
41       # shared volume between host and docker, so that the config file will persist on
40       # the host machine, and you don't have to re-configure everything at each docker startup
39       - ./tempera-accesspoint:/home
38
```

```

async def main():
    try:
        async with asyncio.TaskGroup() as tg:
            _ = tg.create_task(utils.init_globals())
            tempera_station = tg.create_task(bleclient.discovery_loop())
        # Use except* to catch a group (list) of the same exceptions. Must be used when catching exceptions from
        # the async task manager as multiple exceptions can be thrown because of the many concurrent tasks.
        # A simple catch wouldn't allow them all to be handled.
    except* BluetoothOffException:
        logger.critical("Bluetooth is off. Turn on Bluetooth and try again :)")
        sys.exit(0)
    except* RuntimeError:
        logger.info("Returning to device discovery.")
        raise
```

Tempera

Search docs

CONTENTS

BLE app

Configuration script

BLE application

Logfiles and automatic restart at reboot

Tempera API docs

Tempera API docs

API

BLE functionality

Tempera station connection

ETL pipeline

Database

Utilities

BLE app

View page source

BLE app

Configuration script

A python configuration script `configure.py` is provided to configure the parameters necessary for running the BLE application.

Warning

Setting the parameters by manually writing/changing the `conf.yaml` parameter file can easily cause problems or program malfunction. Use the provided script.

Parameters:

- provider – the client (when searching services), else the service (when searching for characteristics).
- uuid – the UUID substring to find.

Returns:

the service or characteristic that has the passed uuid as substring.

```
async tempera.bleclient.etl.measurements_handler(client: BleakClient, characteristics: List[BleakGATTCharacteristic])
```

Read the measurement service characteristics from the device and create/save the resulting measurement to db.

Parameters:

- client – the connection to the tempera station.
- characteristics – list of measurement characteristics to read from the tempera station.

Returns:

None

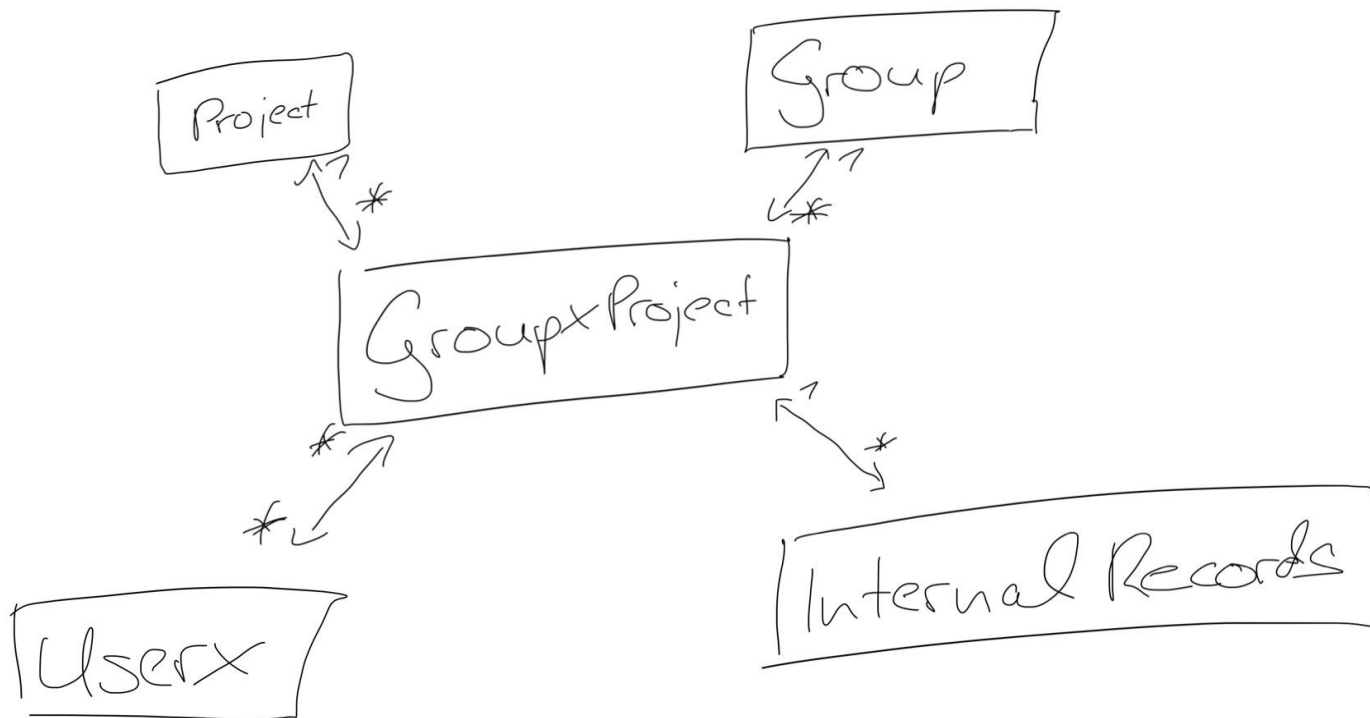
Raises:

- bleak.exc.BleakError – if connection issues occur when reading the characteristics from the tempera station.
- bleak.exc.BleakDBusError – see BleakError



Backend &
Persistence:

Hibernate



```

@Id
@ManyToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE}, fetch = FetchType.EAGER)
private Groupx group;

@Id
@ManyToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE}, fetch = FetchType.EAGER)
private Project project;

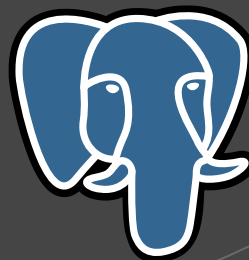
4 usages
@ManyToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE})
private Set<Userx> contributors;

4 usages
@OneToMany(mappedBy = "groupxProject", cascade = CascadeType.ALL)
private Set<InternalRecord> internalRecords;

public GroupxProject() {
    contributors = new HashSet<>();
    internalRecords = new HashSet<>();
}

1 usage
public void addContributor(Userx contributor) {
    contributors.add(contributor);
    contributor.getGroupxProjects().add(this);
}

```



	groupx_projects_group_id	groupx_projects_project_id	contributors_username
1	-1	-1	admin
2	-1	-2	admin
3	-1	-3	admin
4	-1	-4	admin
5	-1	-5	admin
6	-1	-6	admin
7	-2	-7	johndoe
8	-2	-8	johndoe
9	-2	-9	johndoe
10	-2	-10	johndoe
11	-2	-11	johndoe
12	-2	-12	johndoe

1 usage christopherstief

```
@Query("SELECT new at.qe.skeleton.rest.frontend.dtos.SimpleUserDto(u.username, u.firstName, u.lastName, u.email) From GroupxProject gxp JOIN gxp.contributors u where gxp.project.id = :projectId")
public List<SimpleUserDto> findAllContributorsByProject_Id(Long projectId);
```

Software Engineering Tempora / Frontend / loadExtendedProject

Save

GET

{{baseUrl}}/api/project/loadExtendedProject/-6

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

A diagram consisting of a large blue rectangle with a smaller blue rectangle on top of it. The word "Frontend" is written in white text in the center of the large rectangle. The background is dark gray with faint, concentric circles.

Frontend

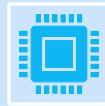
Angular



Angular's component-based structure allows for modular, reusable, and maintainable code.



Ecosystem with built-in tools for routing, state management, form handling, and HTTP communication.



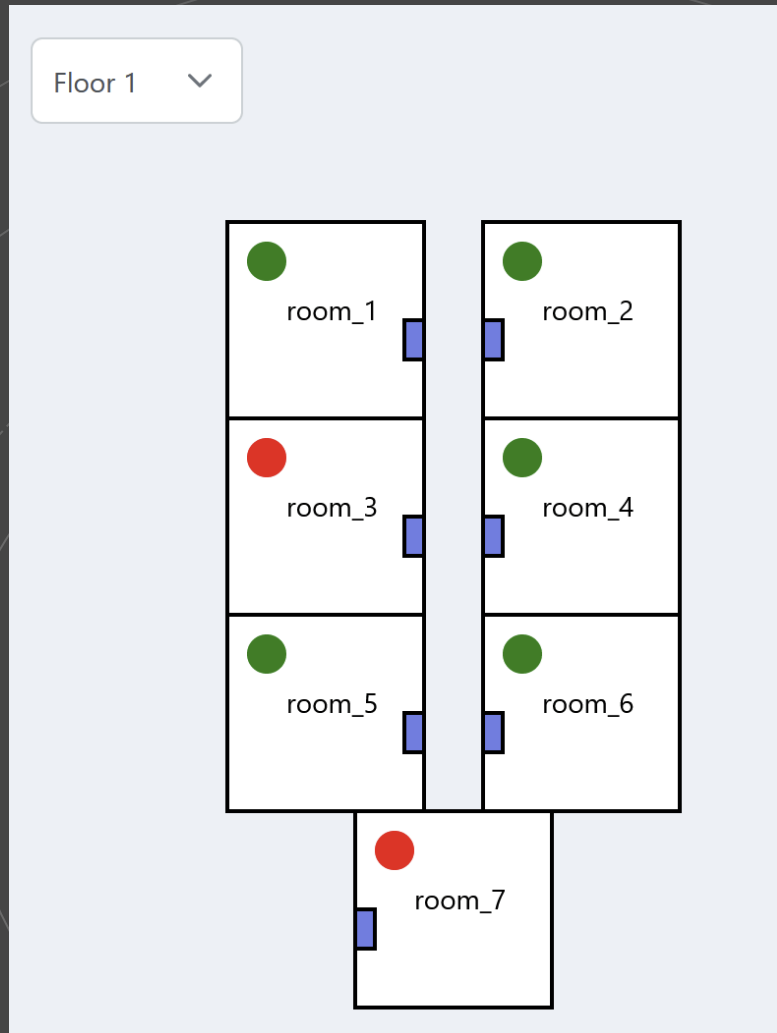
The latest version of Angular offers a standalone component feature for even more modular and flexible application design.



PrimeNG is a popular UI library that offers a wide range of ready-to-use components.



TypeScript allows to catch errors during compilation rather than at runtime.



Floor Plan

Get rooms from backend

Convert to SVG diagram

Click reference to see details of
room and accesspoint

Change color if accesspoint is not
healthy

```

addMembers() : void {
  from(this.selectedUsers.map(user : User => user.username)) Observable<ObservedValueOf<...>>
    .pipe(concatMap( project: userId : string => this.addMember(userId))) Observable<User>
    .subscribe( observerOrNext: {
      next: response : User => {
        this.loadMembersAndUsers(this.groupId!);
        this.messages = [{severity:'success', summary:'Success', detail:'Members added successfully'}],
        error: err => console.error("Error adding member:", err)
      }
    });
  this.displayAddDialog = false;
  this.selectedUsers = [];
}

```

Asynchronous Operations & Control Flow

- Angular makes it easy to write asynchronous code using features like Observables
- Async pipes can be used to simplify subscription handling
- Understanding control flow in Angular is essential for creating a efficient and responsive applications