TEMPERA

Zeitaufzeichnung und Raumklima

G4T1

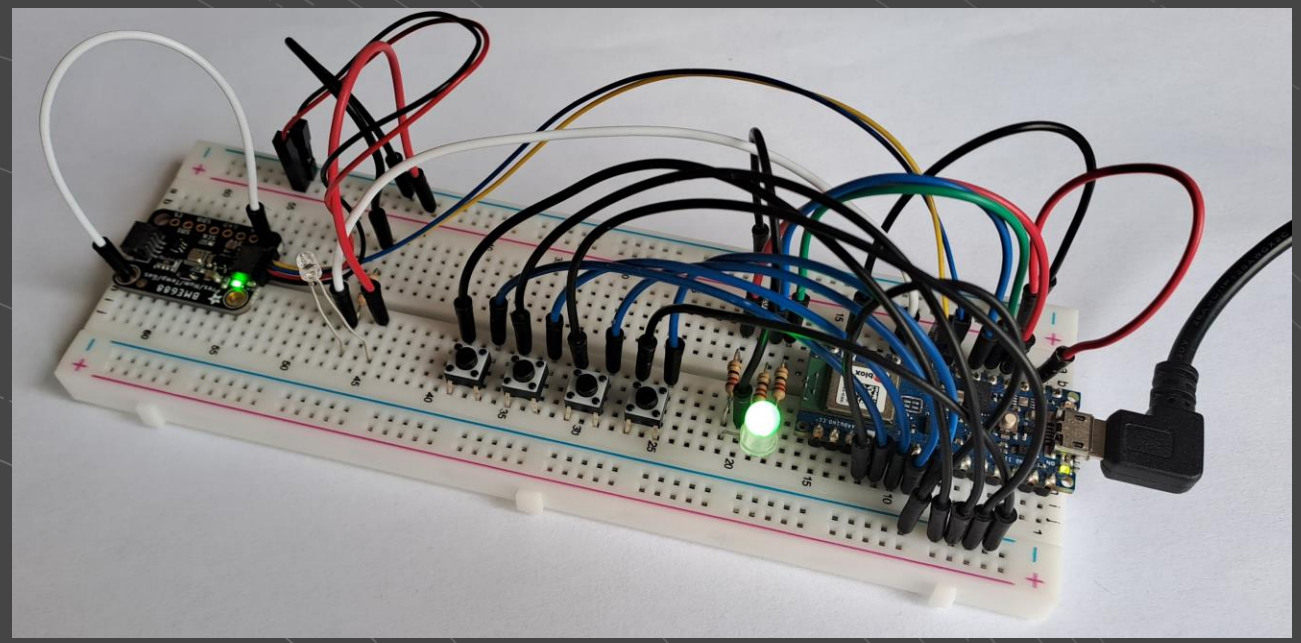
```
// Set LED-color for different work modes (r-g-b
#define DW_COLOR {0, 0, 255}
#define MT_COLOR {255, 40, 10}
#define OO_COLOR {255, 0, 0}
#define PT_COLOR {0, 64, 0}

// Delay in ms after which a new button press wil
#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));
```

```python
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 wh
    # 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

```yaml
services:
  ble-app:
    build: ./tempera-accesspoint
    volumes:
      - /var/run/dbus:/var/run/dbus
      # shared volume between host and docker, so t
      # the host machine, and you don't have to re-
      - ./tempera-accesspoint:/home

  back-end:
    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

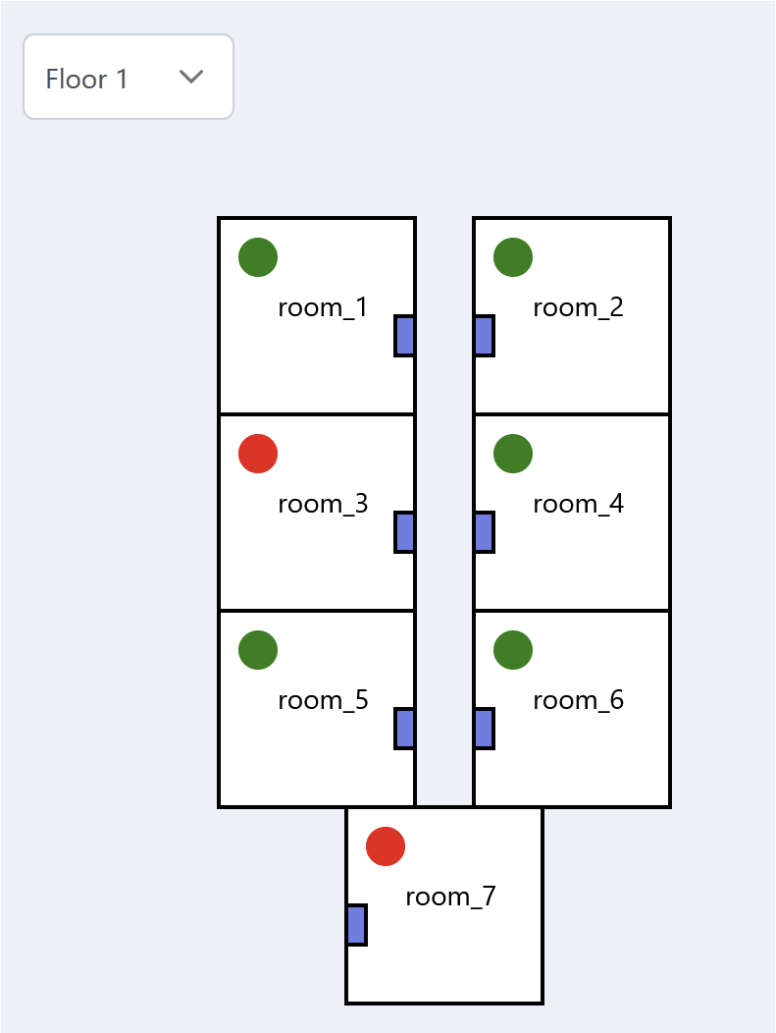
A python configuration script *configure.py* is provided to configure the parameters necessary for running the BLE application.

Warning

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

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 successfull
      },
      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