HTL Rankweil	Laboratorium Raumbezeichnung:	Katalognummer:	25
		Tag der Übung:	17.01.2025
Gruppe: E	Protokoll: Reichart Florian Mitglieder: Mayer Phillip	Klasse	5BHEL

Lehrer	LAP	Titel der Übung **Bluetooth**	Übungsnummer	2
Geprüft			Abgabe am	24.01.2025

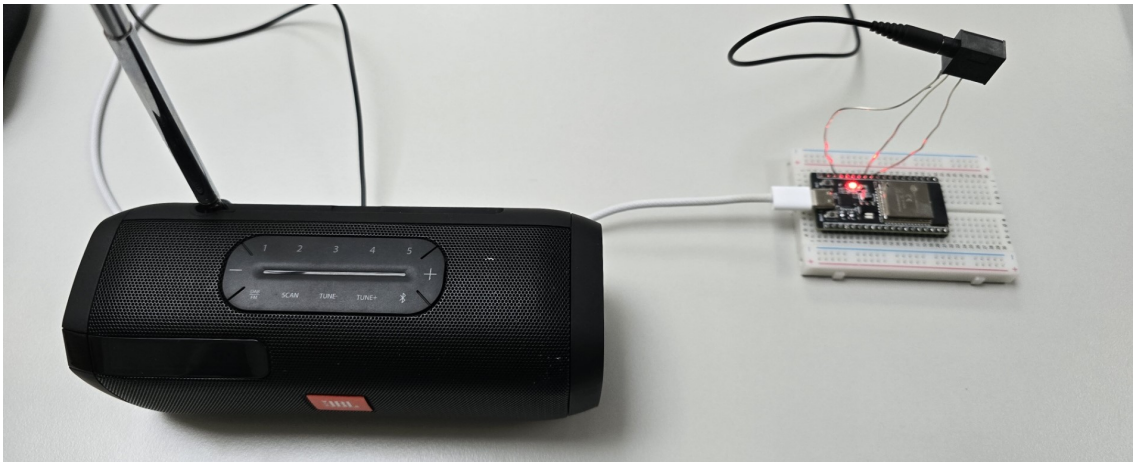
Inhaltsverzeichnis

1 Aufgabe 1: Bluetooth-Lautsprecher

1.1 Aufgabenstellung

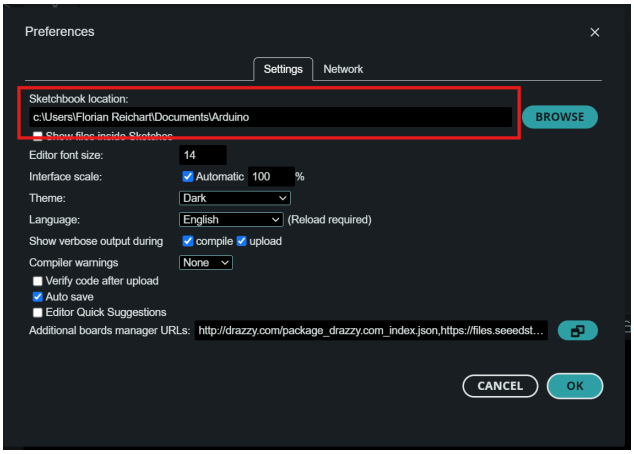
In dieser Aufgabe soll der ESP32 als Bluetooth-Lautsprecher fungieren. Dafür soll die Arduino IDE mit der fertigen Bibliothek von pschatzmann (Quelle) verwendet werden. Um die Funktion zu überprüfen, soll ein Lautsprecher an den ESP32 angeschlossen werden.

1.2 Aufbau

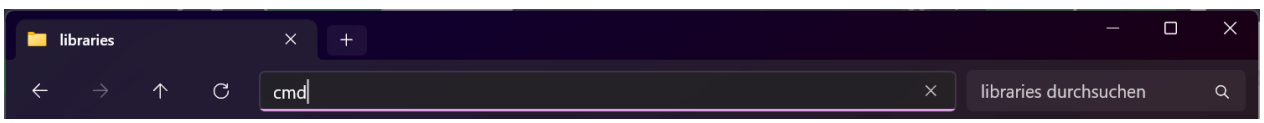


1.3 Installation der Bibliothek

Im Readme der Bibliothek wird die Installation beschrieben. Zuerst muss man den *libraries*-Ordner der Arduino IDE öffnen. Dieser befindet sich standardmäßig unter *Documents/Arduino/libraries* und kann bei Bedarf auch in der Arduino IDE ausgelesen werden. Nachdem dieser Ordner geöffnet wurde, müssen in



ihm zwei Befehle ausgeführt werden. Wenn der Ordner im Windows-Explorer geöffnet wurde, kann in der Adressleiste der Befehl *cmd* eingegeben werden.



Im Befehlsfenster müssen nun folgende Befehle eingegeben werden:

```
git clone https://github.com/pschatzmann/ESP32-A2DP.git
git clone https://github.com/pschatzmann/arduino-audio-tools.git
```

Somit sollten die Bibliotheken installiert und nutzbar sein.

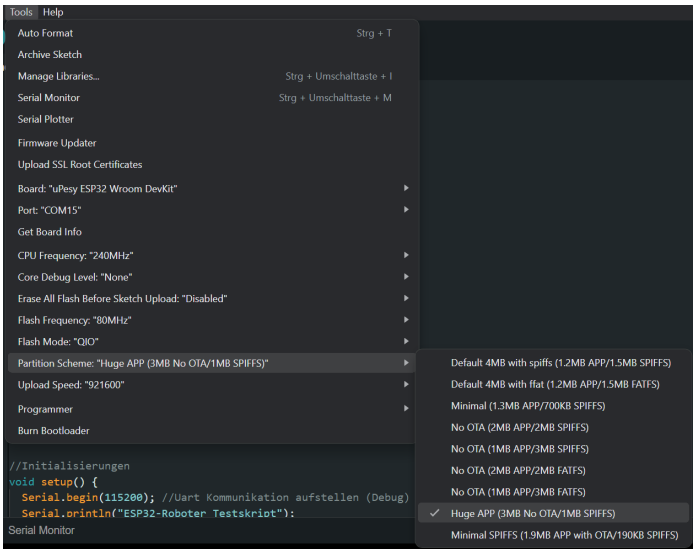
1.4 Programmcode

Der folgende Programmcode stammt aus dem GitHub der Bibliothek.

```
#include "AudioTools.h"
#include "BluetoothA2DPSink.h"

AnalogAudioStream out;
BluetoothA2DPSink a2dp_sink(out);

void setup() {
    a2dp_sink.start("ReichartMayer");
}

void loop() {
}
```

1.5 Hochladen des Codes

Um den Code erfolgreich auf den ESP32 hochzuladen, muss das Aufteilungsschema des Speichers geändert werden. Dies kann unter ‘Tools‘ durchgeführt werden.



1.6 Test des Codes

Nachdem der Code hochgeladen wurde, wurde der Lautsprecher mit dem Klinkenstecker mit dem ESP32 verbunden. Anschließend wurde der ESP32 über Bluetooth mit einem Smartphone gekoppelt. Der ESP32 wird als ”ReichartMayerängezeigt, da dies der Startfunktion mitgegeben wurde. Nach der erfolgreichen Kopplung wurde Musik am Smartphone abgespielt, welche am Lautsprecher wiedergegeben wurde.

2 Aufgabe 2: iBeacon mit dem ESP32

2.1 Aufgabenstellung

In dieser Aufgabe soll ein iBeacon mit dem ESP32 erstellt werden. Dieser soll in regelmäßigen Abständen Daten aussenden. Um die Daten zu empfangen, wird die App *nRF Connect* von NORDIC Semiconductor aus dem App Store verwendet.

2.2 Grundlagen von iBeacon

iBeacon ist eine von Apple entwickelte Technologie, die auf Bluetooth Low Energy (BLE) basiert und präzise Standorterkennung in Innenräumen ermöglicht. Dabei senden kleine Sender, sogenannte Beacons, Signale aus, die von kompatiblen Geräten wie Smartphones empfangen werden können. Diese Signale enthalten Informationen, die es Anwendungen erlauben, kontextbezogene Aktionen auszulösen, zum Beispiel das Anzeigen von Angeboten in Geschäften, die Navigation in Gebäuden oder die Automatisierung von Prozessen. Die Technologie zeichnet sich durch ihre Energieeffizienz, einfache Implementierung und vielseitige Einsatzmöglichkeiten im Einzelhandel, in Museen oder auf Messen aus.

2.3 Programmcode

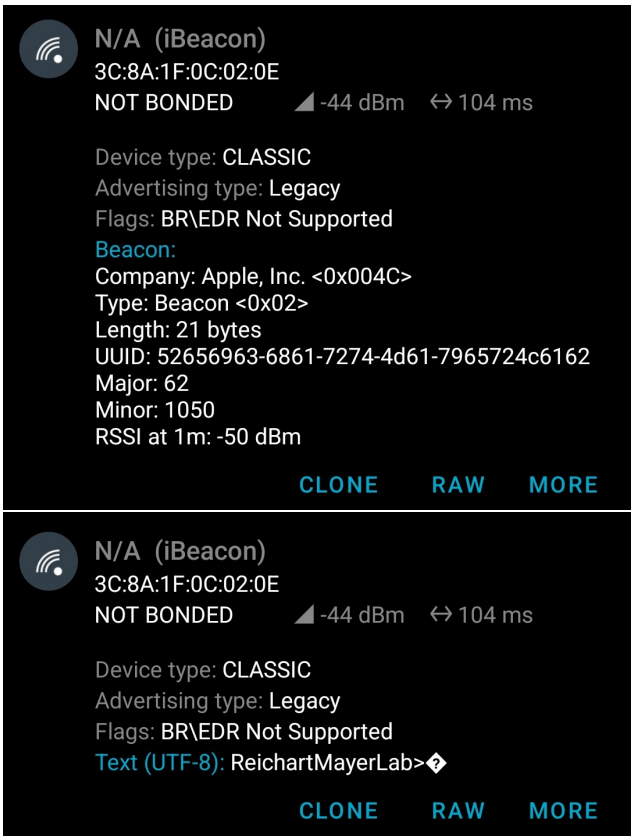
Der folgende Code stammt von N3MIS15 (Quelle). Jedoch wurden die Bytes der uuid angepasst, um im ASCII-Code-Format den String *ReichartMayerLab* anzuzeigen.

```
import struct
import ubluetooth as bt
from micropython import const

MANUFACTURER_ID         = const(0x004C)
DEVICE_TYPE             = const(0x02)
DATA_LENGTH             = const(0x15)
BR_EDR_NOT_SUPPORTED    = const(0x04)
FLAG_BROADCAST          = const(0x01)
MANUFACTURER_DATA       = const(0xFF)

def convert_tx_power(dbm):
    return dbm + 0xFF + 1

class iBeacon():
    def __init__(self, ble, uuid, major, minor, tx_power):
        # Setup BLE
        self.ble = ble
        self.ble.active(False)
        self.ble.active(True)
        print("BLE Activated")

        self.uuid = uuid
        self.major = major
        self.minor = minor
        self.tx_power = convert_tx_power(tx_power)
        self.adv_payload = self.create_payload()


    def create_payload(self):
        payload = bytearray()

```

```
        #Set advertising flag
        value    = struct.pack('B', BR_EDR_NOT_SUPPORTED)
        payload += struct.pack('BB', len(value) + 1, FLAG_BROADCAST) + value

        # Set advertising data
        value    = struct.pack('<H2B', MANUFACTURER_ID, DEVICE_TYPE, DATA_LENGTH)
        value   += self.uuid
        value   += struct.pack(">2HB", self.major, self.minor, self.tx_power)
        payload += struct.pack('BB', len(value) + 1, MANUFACTURER_DATA) + value

        return payload


    def advertise(self, interval_us=100000):
        print("Advertising: " + str(self.adv_payload))
        self.ble.gap_advertise(None)
        self.ble.gap_advertise(interval_us, adv_data=self.adv_payload,
        ↪  connectable=False)


    def update(self, major, minor, advertise_interval):
        self.ble.active(False)

        self.major = major
        self.minor = minor
        self.adv_payload = self.create_payload()

        self.ble.active(True)
        self.advertise(advertise_interval)


def demo():
    beacon = iBeacon(
        ble         = bt.BLE(),
        uuid        = bytearray((
                        0x52, 0x65, 0x69, 0x63, 0x68, 0x61, 0x72, 0x74,
                        0x4d, 0x61, 0x79, 0x65, 0x72, 0x4c, 0x61, 0x62
                    )),
        major       = 62,
        minor       = 1050,
        tx_power    = -50,
    )

    beacon.advertise()

if __name__ == "__main__":
    demo()
    while True:
        pass
```

2.4 Test des Programms

Nachdem der Code auf den ESP32 hochgeladen wurde, wurde die nRF Connect App geöffnet. Dort wurde ein iBeacon-Gerät entdeckt. Nachdem Details über das Gerät angezeigt wurden, wurde erkannt, dass die UUID des Geräts mit der im Code eingestellten übereinstimmt und es sich somit um das selbst programmierte Gerät handelt.



3 Aufgabe 3: Remote ID einer Drohne

3.1 Aufgabenstellung

In dieser Aufgabe soll die Remote ID einer kommerziellen Drohne empfangen und angezeigt werden.

3.2 Programmcode

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>

BLEScan* pBLEScan;
int scanTime = 10;

void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE scan...");
  BLEDevice::init("");
  pBLEScan = BLEDevice::getScan();
  pBLEScan->setActiveScan(true);
  pBLEScan->setInterval(100);
  pBLEScan->setWindow(99);
  pBLEScan->start(scanTime, scanComplete);
}

void loop() {
  delay(1000);
}

void scanComplete(BLEScanResults scanResults) {
  Serial.print("Scan complete, found ");
  Serial.print(scanResults.getCount());
  Serial.println(" devices");

  for (int i = 0; i < scanResults.getCount(); i++) {
    BLEAdvertisedDevice device = scanResults.getDevice(i);
    Serial.print("Device Name: ");
    Serial.println(device.getName().c_str());
    Serial.print("Device Address: ");
    Serial.println(device.getAddress().toString().c_str());
    Serial.print("RSSI: ");
    Serial.println(device.getRSSI());
    Serial.print("Tx Power: ");
    Serial.println(device.getTXPower());
    Serial.print("Service UUIDs: ");
    String serviceUUIDs = device.getServiceUUID().toString();
    Serial.println(serviceUUIDs.c_str());
    Serial.print("Manufacturer Data: ");
    if (device.haveManufacturerData()) {
      String manufacturerData = device.getManufacturerData();
      Serial.println(manufacturerData.c_str());
    }
```

```
    Serial.print("Service Data: ");
48  if (device.haveServiceData()) {
      String serviceData = device.getServiceData();
50    Serial.println(serviceData.c_str());
    }
52  Serial.print("Appearance: ");
    Serial.println(device.getAppearance());
54  Serial.println();
  }
56 }
```

3.2.1 Test des Programms

Das Programm wurde ausgeführt und verschiedene Daten, inklusive der Remote ID, wurden von den Geräten angezeigt. Somit kann diese auch von kommerziellen Drohnen und anderen Geräten ausgelesen werden.

4 Aufgabe 4: Dokumentation

4.1 Bearbeitung von LaTeX

Latex erzeugt eine PDF, die zur visuellen Kontrolle benutzt werden kann. Um den reinen Latex-Code anzusehen und zu bearbeiten, kann ein Latex-Editor wie Overleaf genutzt werden. Nachdem dort ein Konto erstellt wurde, kann das Projekt über *New Project* → *Upload Project* als ZIP importiert werden.