# BLE - Uart

## Übersicht

Mit dem ESP32 lässt sich ein BLE Uart sehr einfach realisieren. Die eigentlich Bibliothek funktioniert nicht daher gibt es eine interne eigene Version:

esp32_ble_arduino_id1841.zip

Original : https://github.com/nkolban/ESP32_BLE_Arduino

### Beispiel Code

example.cpp

```
#include "Arduino.h"
/*
    Video: https://www.youtube.com/watch?v=oCMOYS71NIU
    Based on Neil Kolban example for IDF:
https://github.com/nkolban/esp32-snippets/blob/master/cpp_utils/tests/BLE%20Tests/SampleNotify.cpp
    Ported to Arduino ESP32 by Evandro Copercini

   Create a BLE server that, once we receive a connection, will send
periodic notifications.
   The service advertises itself as: 6E400001-B5A3-F393-E0A9-
E50E24DCCA9E
   Has a characteristic of: 6E400002-B5A3-F393-E0A9-E50E24DCCA9E - used
for receiving data with "WRITE"
   Has a characteristic of: 6E400003-B5A3-F393-E0A9-E50E24DCCA9E - used
to send data with  "NOTIFY"

   The design of creating the BLE server is:
   1. Create a BLE Server
   2. Create a BLE Service
   3. Create a BLE Characteristic on the Service
   4. Create a BLE Descriptor on the characteristic
   5. Start the service.
   6. Start advertising.

   In this example rxValue is the data received (only accessible inside
that function).
   And txValue is the data to be sent, in this example just a byte
incremented every second.
*/
#include <BLEDevice.h>
```

```
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
uint8_t txValue = 0;

// See the following for generating UUIDs:
// https://www.uuidgenerator.net/

#define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
// UART service UUID
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"


class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};

class MyCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
      std::string rxValue = pCharacteristic->getValue();
      char myArray[rxValue.size()+1];//as 1 char space for null is also
required
      strcpy(myArray, rxValue.c_str());
      if (rxValue.length() > 0) {
        Serial.println("*********");
        Serial.print("Received Value: ");
        for (int i = 0; i < sizeof(myArray) - 1; i++)
          Serial.print(myArray[i]);

        Serial.println();
        Serial.println("*********");
      }
    }
};


void setup() {
  Serial.begin(115200);

  // Create the BLE Device
```

```
  BLEDevice::init("UART Service");

  // Create the BLE Server
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  // Create the BLE Service
  BLEService *pService = pServer->createService(SERVICE_UUID);

  // Create a BLE Characteristic
  pCharacteristic = pService->createCharacteristic(
                      CHARACTERISTIC_UUID_TX,
                      BLECharacteristic::PROPERTY_NOTIFY
                    );

  pCharacteristic->addDescriptor(new BLE2902());

  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID_RX,
BLECharacteristic::PROPERTY_WRITE
                                       );

  pCharacteristic->setCallbacks(new MyCallbacks());

  // Start the service
  pService->start();

  // Start advertising
  pServer->getAdvertising()->start();
  Serial.println("Waiting a client connection to notify...");
}

void loop() {

  if (deviceConnected) {
    Serial.printf("*** Sent Value: %d ***\n", txValue);
    //pCharacteristic->setValue(&txValue, 1);
    std::string temp = "cat\n";
    pCharacteristic->setValue(temp);
    pCharacteristic->notify();
    txValue++;
  }
  delay(1000);
}
```