Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Zavod za automatiku i računalno inženjerstvo

Sveprisutno računarstvo

4. Laboratorijska vježba

Bluetooth Low Energy

Bluetooth Low Energy je varijanta tehnologije Bluetooth koja štedi energiju, a dizajnirana je za korištenje od strane IoT uređaja. ESP-IDF trenutno podržava dva Bluetooth stoga. Stog temeljen na Bluedroidu podržava klasični Bluetooth kao i BLE. S druge strane, stog baziran na Apache NimBLE je samo BLE. U ovoj vježbi se koristi NimBLE [1].

nRF Connect

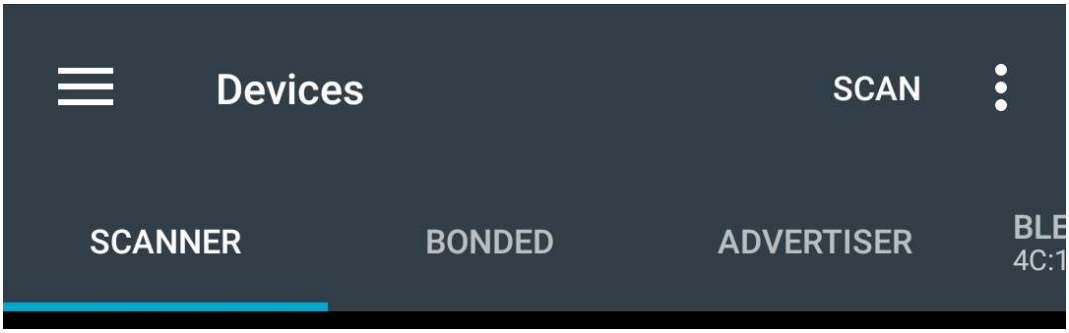
U sklopu vježbe se koristi nRF Connect aplikacija. Aplikacija se može preuzeti za razne platforme.

Desktop: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop/Download

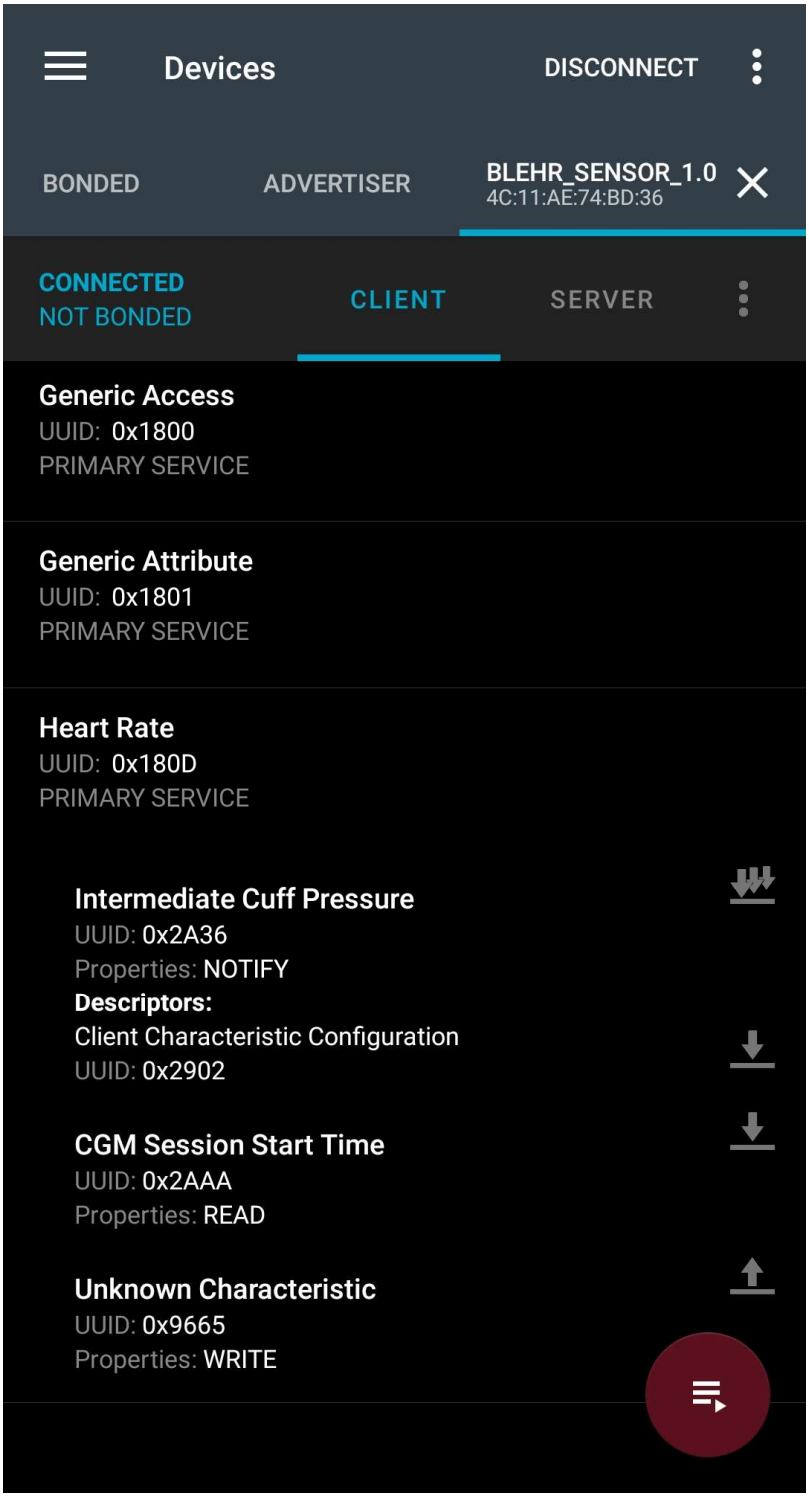
Android: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

iOS: https://apps.apple.com/us/app/nrf-connect-for-mobile/id1054362403

Nakon pokretanja aplikacije odlaskom na karticu SCANNER i pritiskom na tipku SCAN ispisuju se Bluetooth uređaji u dometu.



Klikom na CONNECT se uspostavlja veza i otvara nova kartica za povezani uređaj. Moguće je vidjeti sve servise i karakteristike povezanog uređaja. Klikom na servis su vidljive i karakteristike s kojima postoji interakcija pritiskom na tipke desno od karakteristike.



Primjer

Jednostavno korištenje NimBLE API-a:

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
```

```
#include "freertos/event_groups.h"
#include "esp_event.h"
#include "nvs_flash.h"
#include "esp_log.h"
#include "esp_nimble_hci.h"
#include "nimble/nimble_port.h"
#include "nimble/nimble_port_freertos.h"
#include "host/ble_hs.h"
#include "services/gap/ble_svc_gap.h"
#include "services/gatt/ble_svc_gatt.h"
#include "sdkconfig.h"

char *TAG = "BLE-Server";
uint8_t ble_addr_type;
void ble_app_advertise(void);

static int device_write(uint16_t conn_handle, uint16_t attr_handle,
struct ble_gatt_access_ctxt *ctxt, void *arg){
    printf("Data from the client: %.*s\n", ctxt->om->om_len, ctxt->om-
>om_data);
    return 0;
}

static int device_read(uint16_t con_handle, uint16_t attr_handle, struct
ble_gatt_access_ctxt *ctxt, void *arg){
    os_mbuf_append(ctxt->om, "Data from the server", strlen("Data from
the server"));
    return 0;
}

static const struct ble_gatt_svc_def gatt_svcs[] = {
    {.type = BLE_GATT_SVC_TYPE_PRIMARY,
     .uuid = BLE_UUID16_DECLARE(0x180),                // Define UUID
for device type
     .characteristics = (struct ble_gatt_chr_def[]){
         {.uuid = BLE_UUID16_DECLARE(0xFEF4),          // Define UUID
for reading
          .flags = BLE_GATT_CHR_F_READ,
          .access_cb = device_read},
         {.uuid = BLE_UUID16_DECLARE(0xDEAD),          // Define UUID
for writing
          .flags = BLE_GATT_CHR_F_WRITE,
          .access_cb = device_write},
         {0}}},
    {0}};
```

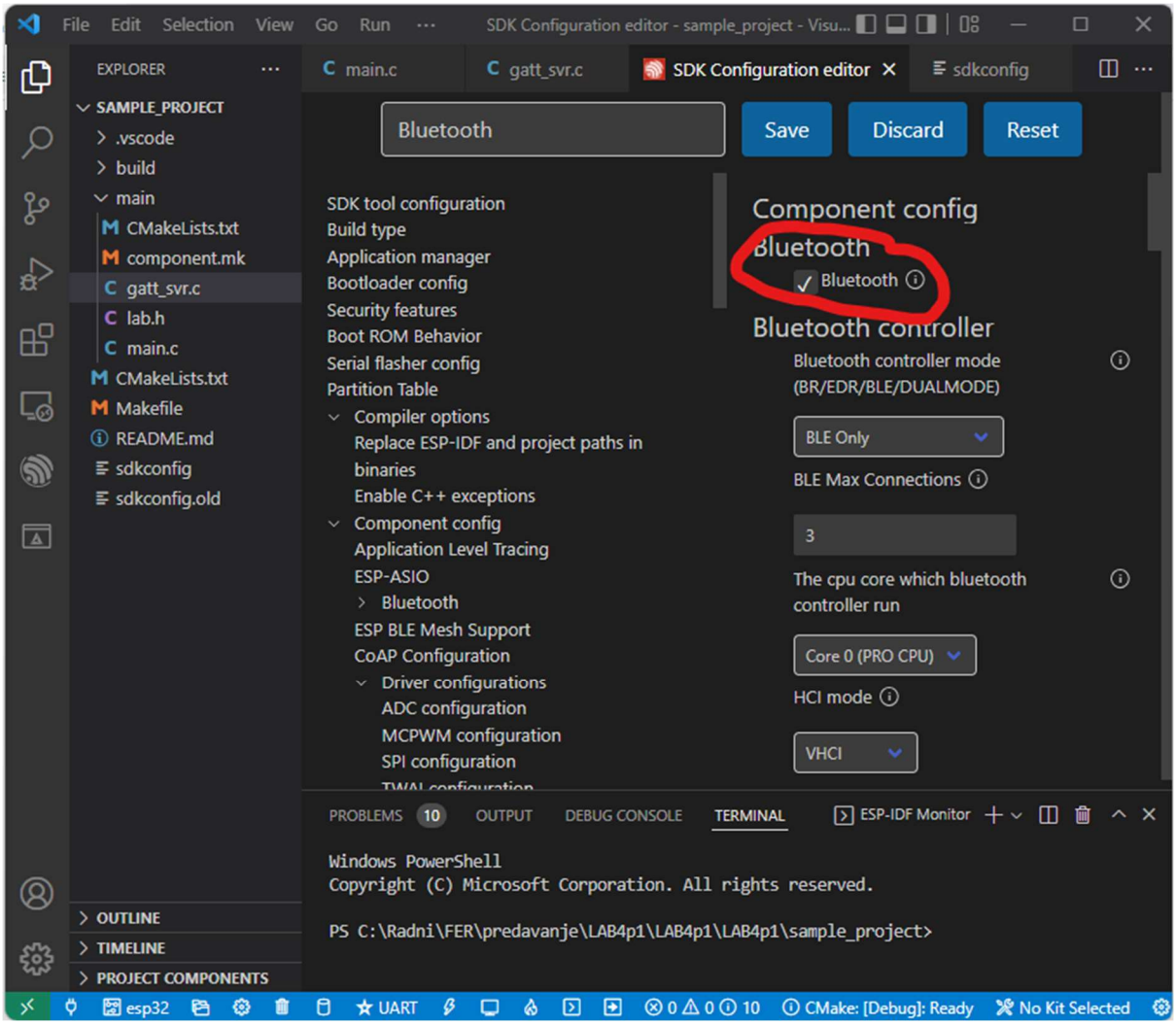
```
// BLE event handling
static int ble_gap_event(struct ble_gap_event *event, void *arg){
    switch (event->type){
    // Advertise if connected
    case BLE_GAP_EVENT_CONNECT:
        ESP_LOGI("GAP", "BLE GAP EVENT CONNECT %s", event-
>connect.status == 0 ? "OK!" : "FAILED!");
        if (event->connect.status != 0){
            ble_app_advertise();
        }
        break;
    // Advertise again after completion of the event
    case BLE_GAP_EVENT_ADV_COMPLETE:
        ESP_LOGI("GAP", "BLE GAP EVENT");
        ble_app_advertise();
        break;
    default:
        break;
    }
    return 0;
}

// Define the BLE connection
void ble_app_advertise(void){
    struct ble_hs_adv_fields fields;
    const char *device_name;
    memset(&fields, 0, sizeof(fields));
    device_name = ble_svc_gap_device_name();
    fields.name = (uint8_t *)device_name;
    fields.name_len = strlen(device_name);
    fields.name_is_complete = 1;
    ble_gap_adv_set_fields(&fields);

    // GAP - device connectivity definition
    struct ble_gap_adv_params adv_params;
    memset(&adv_params, 0, sizeof(adv_params));
    adv_params.conn_mode = BLE_GAP_CONN_MODE_UND; // connectable or non-
connectable
    adv_params.disc_mode = BLE_GAP_DISC_MODE_GEN; // discoverable or
non-discoverable
    ble_gap_adv_start(ble_addr_type, NULL, BLE_HS_FOREVER, &adv_params,
ble_gap_event, NULL);
}

void ble_app_on_sync(void){
    ble_hs_id_infer_auto(0, &ble_addr_type); // Determines the best
```

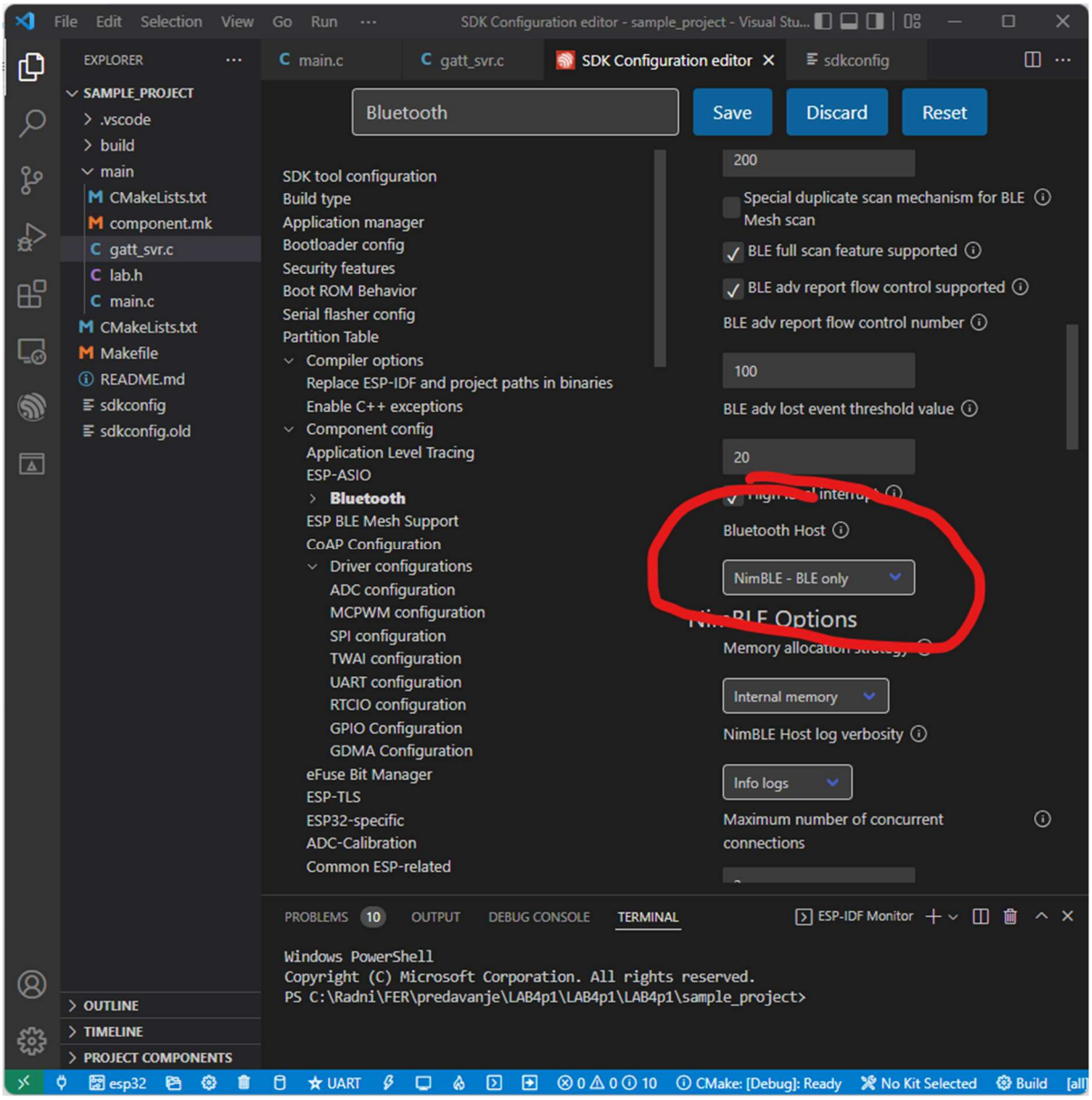
```
address type automatically
    ble_app_advertise();                     // Define the BLE
connection
}

void host_task(void *param){
    nimble_port_run();
}

void app_main(){
    nvs_flash_init();                              // 1 - Initialize
NVS flash using
    esp_nimble_hci_and_controller_init();          // 2 - Initialize
ESP controller
    nimble_port_init();                            // 3 - Initialize
the host stack
    ble_svc_gap_device_name_set("BLE-Server");     // 4 - Initialize
NimBLE configuration - server name
    ble_svc_gap_init();                            // 4 - Initialize
NimBLE configuration - gap service
    ble_svc_gatt_init();                           // 4 - Initialize
NimBLE configuration - gatt service
    ble_gatts_count_cfg(gatt_svcs);                // 4 - Initialize
NimBLE configuration - config gatt services
    ble_gatts_add_svcs(gatt_svcs);                 // 4 - Initialize
NimBLE configuration - queues gatt services.
    ble_hs_cfg.sync_cb = ble_app_on_sync;          // 5 - Initialize
application
    nimble_port_freertos_init(host_task);          // 6 - Run the
thread
}
```

Dodatni primjeri korištenja NimBLE API-a [2].

Napomena: podesiti NimBLE u SDK konfiguraciji.

- *<F1> ESP-IDF: SDK Configuration Editor (menuconfig)*

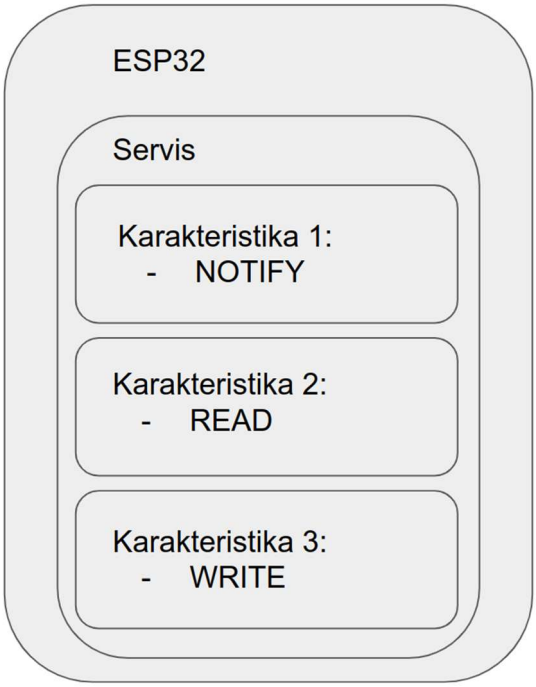

File Edit Selection View Go Run
SDK Configuration editor - sample_project - Visu...
EXPLORER
main.c
gatt_svr.c
SDK Configuration editor
sdkconfig
SAMPLE_PROJECT
.vscode
build
main
CMakeLists.txt
component.mk
gatt_svr.c
lab.h
main.c
CMakeLists.txt
Makefile
README.md
sdkconfig
sdkconfig.old
Bluetooth
Save
Discard
Reset
SDK tool configuration
Build type
Application manager
Bootloader config
Security features
Boot ROM Behavior
Serial flasher config
Partition Table
Compiler options
Replace ESP-IDF and project paths in binaries
Enable C++ exceptions
Component config
Application Level Tracing
ESP-ASIO
Bluetooth
ESP BLE Mesh Support
CoAP Configuration
Driver configurations
ADC configuration
MCPWM configuration
SPI configuration
Component config
Bluetooth
Bluetooth
Bluetooth controller
Bluetooth controller mode (BR/EDR/BLE/DUALMODE)
BLE Only
BLE Max Connections
3
The cpu core which bluetooth controller run
Core 0 (PRO CPU)
HCI mode
VHCI
PROBLEMS 10
OUTPUT
DEBUG CONSOLE
TERMINAL
ESP-IDF Monitor
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Radni\FER\predavanje\LAB4p1\LAB4p1\LAB4p1\sample_project>
OUTLINE
TIMELINE
PROJECT COMPONENTS
esp32
UART
CMake: [Debug]: Ready
No Kit Selected



Zadatak

Prvi dio:

Implementirati BLE GATT server koji poslužuje jedan servis s tri karakteristike. Bluetooth naziv ESP32 je *ImePrezime*. UUID servisa su zadnje 4 znamenke JMBAG-a (*e.g. 0036512345 -> UUID=0x2345*).



Karakteristika 1:

- UUID = (zadnje 4 znamenke JMBAG-a + 1)
- varijabla *counterBLE* (početna vrijednost 0) se svake sekunde povećava za *writeBLE*
- dojava o promjeni varijable mora biti vidljiva u aplikaciji

Karakteristika 2:

- UUID = (zadnje 4 znamenke JMBAG-a + 2)
- na zahtjev (klikom na tipku u aplikaciji) čitanje varijable *readBLE*
- varijabla *readBLE* = JMBAG

Karakteristika 3:

- UUID = (zadnje 4 znamenke JMBAG-a + 3)
- u varijablu *writeBLE* ϵ[1,10] omogućiti zapisivanje broja (klikom na tipku u aplikaciji)

Drugi dio (nije obavezno):

Implementirati BLE GATT klijent koji se može spajati na GATT server. Klijent može čitati vrijednosti karakteristike 3 i pisati nove vrijednosti u karakteristiku 2. Na serijski terminal ispisivati vrijednosti karakteristike 1. Promjena iz GATT servera u klijent se događa na pritisak tipke ili može biti implementirana kao 2 zasebna projekta.

Predaja

Vježbe se predaju preko moodla sukladno uputama koje će biti na web-u.

Reference

[1] NimBLE-based host APIs, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/nimble/index.html

[2] Bluetooth Examples for NimBLE host, https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/nimble