

Umfrage

im Rahmen der Bachelorarbeit
„Reactive Bluetooth Low Energy Framework for iOS“
von Martin Stöber

Auf den folgenden Seiten sind 2 verschiedene Implementierungen abgebildet. Bitte schauen Sie sich diese an und wählen Sie für jede der unten abgebildeten Kategorien eine der beiden Implementierungen aus, indem Sie die entsprechende Zelle mit einem X markieren.

Die Implementierungen stellen eine Bluetooth Kommunikation dar. Dabei interagieren 2 Geräte miteinander. Der Peripheral (vergleichbar mit einem Server) stellt ein Datenschema zur Verfügung. Der Central (vergleichbar mit einem Client) verbindet sich mit diesem, liest den Wert aus und setzt einen neuen Wert.

Weitere Fragen können persönlich geklärt werden.

	Standard CoreBluetooth Implementierung	Reactive Bluetooth LE Framework for iOS Implementierung
Verständlichkeit		
Änderbarkeit		
Testbarkeit		
Generelle Bevorzugung		

Teil I.

CoreBluetooth Implementierung

Listing 1: CoreBluetooth Peripheral Implementierung

```
1 #import <CoreBluetooth/CoreBluetooth.h>
2
3
4 @interface CoreBluetoothImplementationPeripheral : NSObject <CBPeripheralManagerDelegate>
5
6 @property(n nonatomic) CBPeripheralManager *peripheralManager;
7 @property(n nonatomic) CBMutableService *testService;
8 @property(n nonatomic) CBMutableCharacteristic *testCharacteristic;
9
10 @end
11
12
13 @implementation CoreBluetoothImplementationPeripheral
14
15 - (instancetype)init {
16     self = [super init];
17     if (self) {
18         _peripheralManager = [[CBPeripheralManager alloc] initWithDelegate:self queue:nil];
19         [self setup];
20     }
21     return self;
22 }
23
24 - (void)setup {
25     CBUUID *testServiceUUID = [CBUUID UUIDWithString:@"AAAAAAAA-AA-AAA-AAA-AAAAAAAAAAAA"];
26     CBUUID *testCharacteristicUUID = [CBUUID UUIDWithString:@"BBBBBBB-BBBB-BBBB-BBBB-BBBBBBBBBBBB"];
27
28     // create attributes structure
29     self.testService = [[CBMutableService alloc] initWithType:testServiceUUID primary:YES];
30     self.testCharacteristic = [[CBMutableCharacteristic alloc] initWithType:testCharacteristicUUID
31         properties:(CBCharacteristicPropertyRead | CBCharacteristicPropertyWrite) value:nil permissions:(
32             CBAAttributePermissionsReadable | CBAAttributePermissionsWriteable)];
33     self.testService.characteristics = @[self.testCharacteristic];
34 }
35
36 - (void)peripheralManagerDidUpdateState:(CBPeripheralManager *)peripheral {
37     if (peripheral.state == 5) {
38         // publish attributes
39         [self.peripheralManager addService:self.testService];
40     }
41 }
42
43 - (void)peripheralManager:(CBPeripheralManager *)peripheral didAddService:(CBService *)service error:(
44     NSError *)error {
45     if (error) {
46         return;
47     }
48
49     // set characteristic value
50     self.testCharacteristic.value = @"TEST-VALUE" dataUsingEncoding:NSUTF8StringEncoding];
51
52     // set device name & start advertising
53     [self.peripheralManager startAdvertising:@{CBAdvertisementDataLocalNameKey : @"TEST-DEVICE",
54         CBAdvertisementDataServiceUUIDsKey : @[self.testService.UUID]}];
55 }
56
57 // read request handling
58 - (void)peripheralManager:(CBPeripheralManager *)peripheral didReceiveReadRequest:(CBATTRequest *)request
59 {
60     if (request.offset > self.testCharacteristic.value.length) {
61         [self.peripheralManager respondToRequest:request withResult:CBATTErrorInvalidOffset];
62         return;
63     }
64     request.value = [self.testCharacteristic.value subdataWithRange:NSMakeRange(request.offset,
65         self.testCharacteristic.value.length - request.offset)];
66     [self.peripheralManager respondToRequest:request withResult:CBATTErrorSuccess];
67 }
68
69 // write request handling
70 - (void)peripheralManager:(CBPeripheralManager *)peripheral didReceiveWriteRequests:(NSArray *)requests {
71     for (CBATTRequest *request in requests) {
72         self.testCharacteristic.value = request.value;
73         [self.peripheralManager respondToRequest:request withResult:CBATTErrorSuccess];
74     }
75 }
76
77 @end
```

Listing 2: CoreBluetooth Central Implementierung

```
1 #import <CoreBluetooth/CoreBluetooth.h>
2
3
4 @interface CBSpeedTester : NSObject <CBCentralManagerDelegate, CBPeripheralDelegate>
5
6 @property(n nonatomic) CBCentralManager *centralManager;
7 @property(n nonatomic) CBPeripheral *peripheral;
8 @property(n nonatomic) CBService *service;
9 @property(n nonatomic) CBCharacteristic *characteristic;
10
11 @end
12
13
14 @implementation CBSpeedTester
15
16
17 - (instancetype)init {
18     self = [super init];
19     if (self) {
20         _centralManager = [[CBCentralManager alloc] initWithDelegate:self queue:dispatch_get_main_queue()
21                             ];
22     }
23     return self;
24 }
25
26 - (void)centralManagerDidUpdateState:(CBCentralManager *)central {
27     if (central.state == 5) {
28         self.startDate = [NSDate date];
29
30         // scan for peripherals
31         [self.centralManager scanForPeripheralsWithServices:nil options:nil];
32     }
33 }
34
35 - (void)centralManager:(CBCentralManager *)central didDiscoverPeripheral:(CBPeripheral *)peripheral
36     advertisementData:(NSDictionary *)advertisementData RSSI:(NSNumber *)RSSI {
37     if (RSSI.integerValue > -15) {
38         return;
39     }
40
41     self.peripheral = peripheral;
42     self.peripheral.delegate = self;
43     [central stopScan];
44
45     // connect to peripheral
46     [central connectPeripheral:peripheral options:nil];
47 }
48
49 - (void)centralManager:(CBCentralManager *)central didConnectPeripheral:(CBPeripheral *)peripheral {
50     if (peripheral == self.peripheral) {
51
52         // discover services
53         [self.peripheral discoverServices:nil];
54     }
55 }
56
57 - (void)peripheral:(CBPeripheral *)peripheral didDiscoverServices:(NSError *)error {
58     if (!error && peripheral == self.peripheral) {
59         CBUUID *testServiceUUID = [CBUUID UUIDWithString:@"AAAAAAAA-AAAA-AAAA-AAAA-AAAAAAAAAAAA"];
60         for (CBService *service in peripheral.services) {
61             if ([service.UUID isEqual:testServiceUUID]) {
62                 self.service = service;
63
64                 // discover characteristic
65                 [self.peripheral discoverCharacteristics:nil forService:self.service];
66             }
67         }
68     }
69 }
70
71 - (void)peripheral:(CBPeripheral *)peripheral didDiscoverCharacteristicsForService:(CBService *)service
72     error:(NSError *)error {
73     if (!error && service == self.service) {
74         CBUUID *testCharacteristicUUID = [CBUUID UUIDWithString:@"BBBBBBBB-BBBB-BBBB-BBBB-BBBBBBBBBBBB"];
75         for (CBCharacteristic *characteristic in self.service.characteristics) {
76             if ([characteristic.UUID isEqual:testCharacteristicUUID]) {
77                 self.characteristic = characteristic;
78             }
79         }
80     }
81 }
```

```
76
77         // read characteristic
78         [self.peripheral readValueForCharacteristic:self.characteristic];
79     }
80 }
81 }
82 }
83
84 - (void)peripheral:(CBPeripheral *)peripheral didUpdateValueForCharacteristic:(CBCharacteristic *)
85     characteristic error:(NSError *)error {
86     if (!error) {
87         NSLog(@"Characteristic Value: %@", characteristic.value);
88
89         // write characteristic
90         [self.peripheral writeValue:@"Test" dataUsingEncoding:NSUTF8StringEncoding] forCharacteristic:
91             self.characteristic type:CBCharacteristicWriteWithResponse];
92     }
93 }
94
95 - (void)peripheral:(CBPeripheral *)peripheral didWriteValueForCharacteristic:(CBCharacteristic *)
96     characteristic error:(NSError *)error {
97     if (!error && characteristic == self.characteristic) {
98
99         // disconnect peripheral
100         [self.centralManager cancelPeripheralConnection:self.peripheral];
101     }
102 }
103
104 @end
```

Teil II.

Reactive Bluetooth LE Framework for iOS Implementierung

Listing 3: Reactive Peripheral Implementierung

```
1 #import <ReactiveBluetoothLE/ReactiveBluetoothLE.h>
2
3
4 @interface ReactiveImplementationPeripheral : NSObject
5
6 @property(n nonatomic) RBTPeripheralModule *peripheralModule;
7 @property(n nonatomic) RBTMutableService *testService;
8 @property(n nonatomic) RBTMutableCharacteristic *testCharacteristic;
9
10 @end
11
12
13 @implementation ReactiveImplementationPeripheral
14
15 - (instancetype)init {
16     self = [super init];
17     if (self) {
18         _peripheralModule = [[RBTPeripheralModule alloc] init];
19         [self setup];
20     }
21     return self;
22 }
23
24 - (void)setup {
25     CBUUID *testServiceUUID = [CBUUID UUIDWithString:@"AAAAAAAA-AAAA-AAAA-AAAA-AAAAAAAAAAAA"];
26     CBUUID *testCharacteristicUUID = [CBUUID UUIDWithString:@"BBBBBBBB-BBBB-BBBB-BBBB-BBBBBBBBBBBB"];
27
28     // set device name
29     self.peripheralModule.name = @"TEST-DEVICE";
30
31     // create attributes structure
32     self.testService = [[RBTMutableService alloc] initWithPrimaryServiceUUID:testServiceUUID];
33     self.testCharacteristic = [[RBTMutableCharacteristic alloc] initWithUUID:testCharacteristicUUID
34                               properties:(CBCharacteristicPropertyRead | CBCharacteristicPropertyWrite)
35                               value:nil
36                               permissions:(CBAAttributePermissionsReadable | CBAAttributePermissionsWriteable)];
37
38
39     @weakify(self)
40     [[self.peripheralModule.peripheralState filter:^(NSNumber *state) {
41         return state.intValue == 5;
42     }] subscribeNext:^(id x) {
43         @strongify(self)
44
45         // publish attributes
46         [self.testService addCharacteristic:self.testCharacteristic];
47         [[self.peripheralModule addService:self.testService] subscribeCompleted:^(
48             @strongify(self)
49
50             // set characteristic value
51             self.testCharacteristic.value = [@"TEST-VALUE" dataUsingEncoding:NSUTF8StringEncoding];
52
53             // start advertising
54             [self.peripheralModule startAdvertising];
55         )];
56     }];
57 }
58
59 @end
```

Listing 4: Reactive Central Implementierung

```

1  #import <ReactiveBluetoothLE/ReactiveBluetoothLE.h>
2
3  @interface RBTSpeedTester : NSObject
4
5  @property(n nonatomic) RBTCentralModule *centralModule;
6  @property(n nonatomic) RBTPeripheral *peripheral;
7  @property(n nonatomic) RBTService *service;
8  @property(n nonatomic) RBTCharacteristic *characteristic;
9
10 @end
11
12
13 @implementation RBTSpeedTester
14
15 - (instancetype)init {
16     self = [super init];
17     if (self) {
18         _centralModule = [[RBTCentralModule alloc] init];
19         [self initTest];
20     }
21     return self;
22 }
23
24 - (void)initTest {
25     @weakify(self)
26     [[self.centralModule.bluetoothState filter:^(NSNumber *state) {
27         return ([state isEqual:@(5)];
28     }] subscribeNext:^(id x) {
29         @strongify(self)
30         self.startDate = [NSDate date];
31         [self startTest];
32     }];
33 }
34
35 - (void)startTest {
36     CBUUID *testServiceUUID = [CBUUID UUIDWithString:@"AAAAAAAA-AAAA-AAAA-AAAA-AAAAAAAAAAAA"];
37     CBUUID *testCharacteristicUUID = [CBUUID UUIDWithString:@"BBBBBBBB-BBBB-BBBB-BBBB-BBBBBBBBBBBB"];
38     @weakify(self)
39
40     // scan for peripherals
41     [[self.centralModule scan] take:1] subscribeNext:^(RBTPeripheral *peripheral) {
42         @strongify(self)
43         self.peripheral = peripheral;
44
45         // connect to peripheral
46         [[self.peripheral connect] subscribeCompleted:^(
47             @strongify(self)
48
49             // discover services
50             [[self.peripheral discoverServices] subscribeCompleted:^(
51                 @strongify(self)
52                 self.service = [self.peripheral serviceWithUUID:testServiceUUID];
53
54                 // discover characteristics
55                 [[self.service discoverAllCharacteristics] subscribeCompleted:^(
56                     @strongify(self)
57                     self.characteristic = [self.service characteristicWithUUID:testCharacteristicUUID];
58
59                     [self readAndWrite];
60                 }];
61             }];
62     }];
63 }
64 }
65
66 - (void)readAndWrite {
67     @weakify(self)
68
69     // read characteristic
70     [[self.characteristic readValue] subscribeCompleted:^(
71         @strongify(self)
72         NSLog(@"Characteristic Value: %@", self.characteristic.value);
73
74         // write characteristic
75         [[self.characteristic writeValue:@"TEST" dataUsingEncoding:NSUTF8StringEncoding] withResponse:YES
76         ] subscribeCompleted:^(
77             @strongify(self)

```

```
78         // disconnect peripheral
79         [self.peripheral disconnect];
80     }];
81 }];
82 }
83
84 @end
```
