Using Kotlin Coroutines to tame Android Bluetooth® LE

Travis Wyatt

JUUL LABS®

@travisiwyatt

medium.com/juullabs-engineering

APRIL 08+09 DROIDCON BOS19

Android Bluetooth Low Energy

Discover `BluetoothLeScanner`

Connect `BluetoothDevice`

Communicate `BluetoothGatt`

Discover

`BluetoothLeScanner`

Connect

`BluetoothDevice`

Communicate

`BluetoothGatt`

```
// 1. Connect to peripheral (Connect)
// 2. Send "hi" to peripheral (Communicate)
// 1. Connect to peripheral (Connect)
// 2. Discover bluetooth services
// 3. Send "hi" to peripheral (Communicate)
```

```
bluetoothDevice.connectGatt(context, false, callback)
```

```kotlin
val callback = object : BluetoothGattCallback() {

}

bluetoothDevice.connectGatt(context, false, callback)
```

```
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {

  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

```kotlin
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {

}
```

```
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {

  } else {

  }
}
```

```kotlin
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {
    gatt.discoverServices()
  } else {

  }
}
```

```
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {
    gatt.discoverServices()
  } else {
    textView.text = "Connect error"
  }
}
```

```
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {
    gatt.discoverServices()
  } else {
    textView.text = "Connect error"
  }
}
```

```
W/BluetoothGatt: Unhandled exception in callback
    android.view.ViewRootImpl$CalledFromWrongThreadException:
    Only the original thread that created a view hierarchy can touch its views.
        at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7753)
        at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1225)
        at android.view.View.requestLayout(View.java:23093)
        ...
```

```
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {
    gatt.discoverServices()
  } else {
    textView.text = "Connect error"
  }
}
```

```kotlin
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {
    gatt.discoverServices()
  } else {
    runOnUiThread {
      textView.text = "Connect error"
    }
  }
}
```

```
override fun onConnectionStateChange(
  gatt: BluetoothGatt, status: Int, newState: Int
) {
  if (newState == STATE_CONNECTED &&
      status == GATT_SUCCESS
  ) {
    gatt.discoverServices()
  } else {
    runOnUiThread {
      textView.text = "Connect error"
    }
  }
}
```

```kotlin
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {
    if (newState == STATE_CONNECTED &&
        status == GATT_SUCCESS
    ) {
      gatt.discoverServices()
    } else {
      runOnUiThread {
        textView.text = "Connect error"
      }
    }
  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

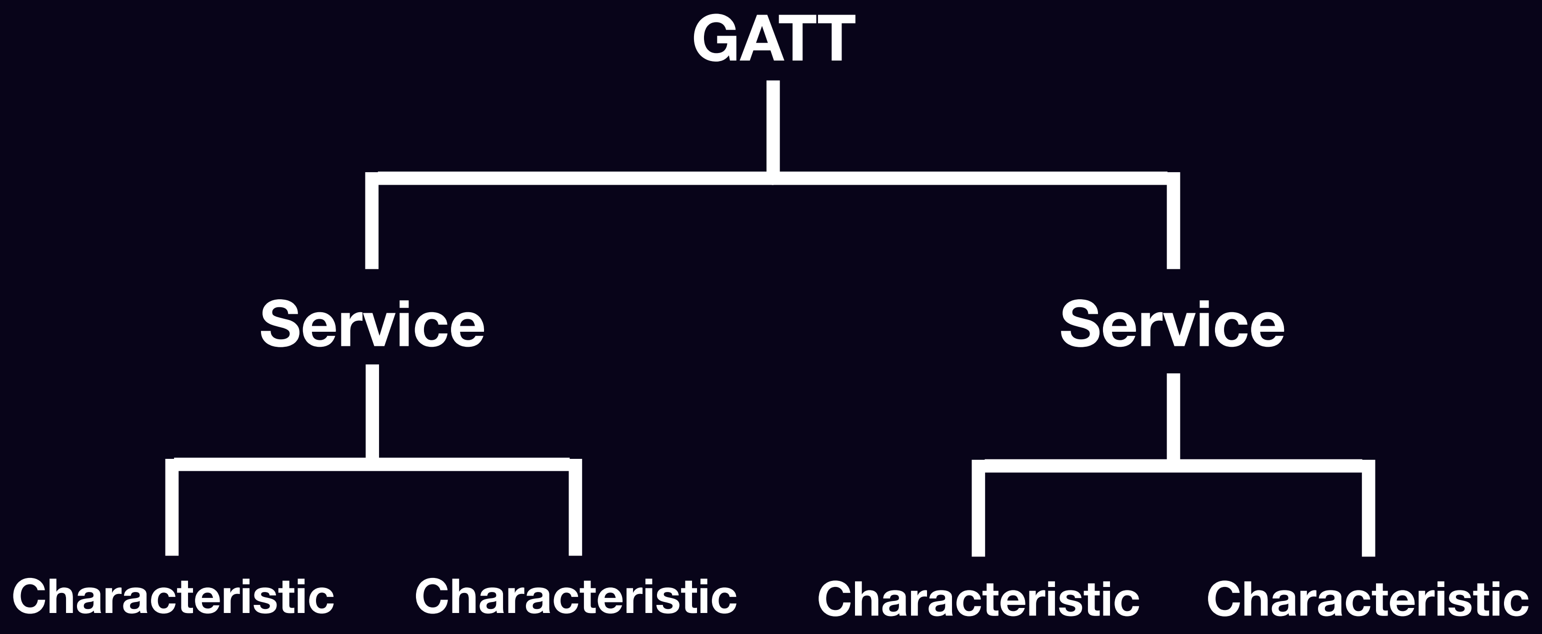
```
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {
    if (newState == STATE_CONNECTED &&
        status == GATT_SUCCESS
    ) {
      gatt.discoverServices()
    } else {
      runOnUiThread {
        textView.text = "Connect error"
      }
    }
  }

  override fun onServicesDiscovered(
    gatt: BluetoothGatt, status: Int
  ) {

  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {

}
```

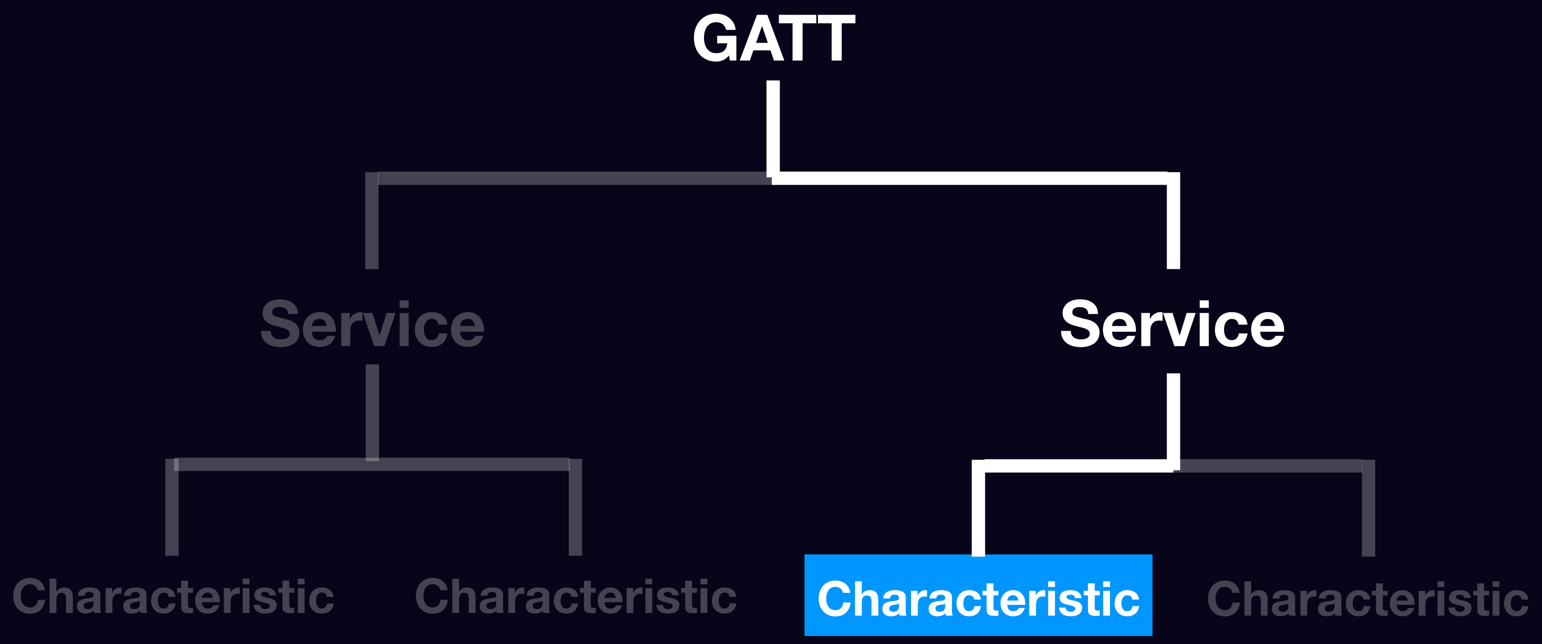
```kotlin
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {

  } else {

  }
}
```

```kotlin
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
  } else {

  }
}
```

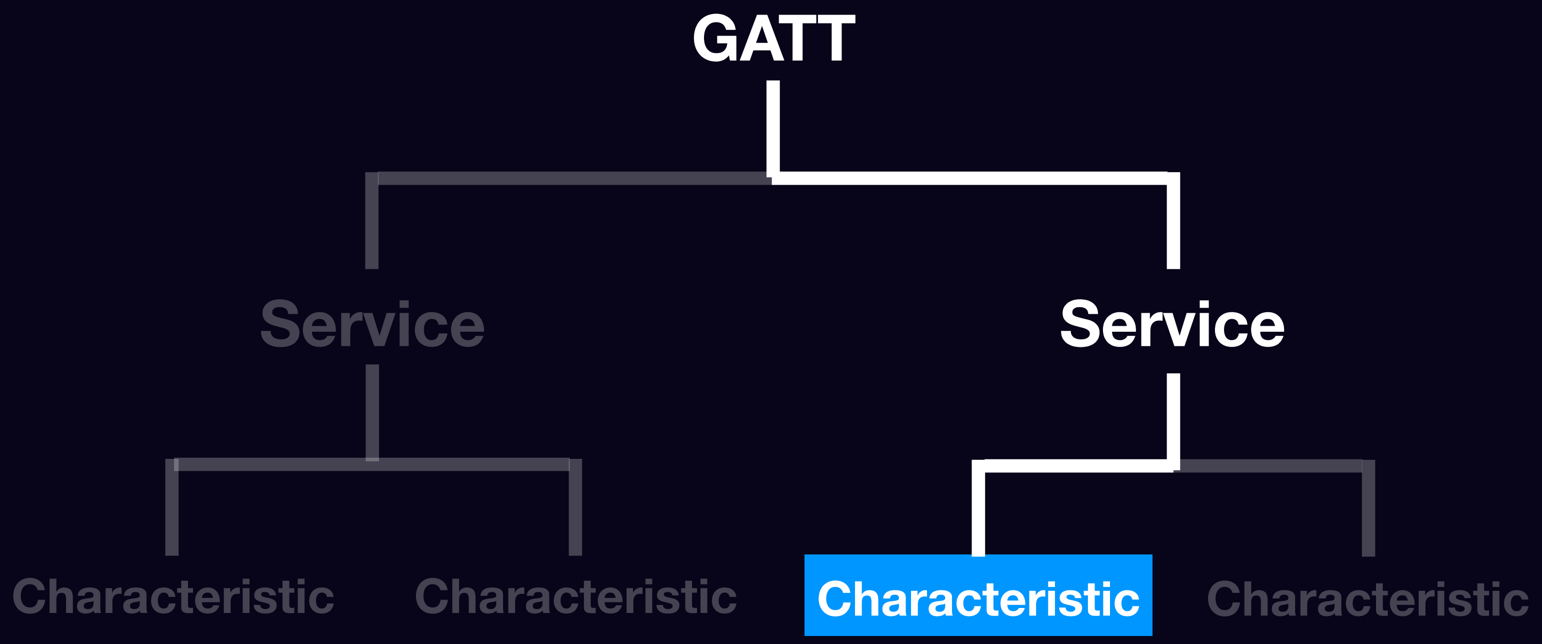
```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
  } else {

  }
}
```



```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
  } else {

  }
}
```



```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
    characteristic.value = "hi".toByteArray()
    gatt.writeCharacteristic(characteristic)
  } else {

  }
}
```



```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
    characteristic.value = "hi".toByteArray()
    gatt.writeCharacteristic(characteristic)
  } else {
    textView.text = "Discover error"
  }
}
```

```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
    characteristic.value = "hi".toByteArray()
    gatt.writeCharacteristic(characteristic)
  } else {
    textView.text = "Discover error"
  }
}
```

```
W/BluetoothGatt: Unhandled exception in callback
    android.view.ViewRootImpl$CalledFromWrongThreadException:
    Only the original thread that created a view hierarchy can touch its views.
        at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7753)
        at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1225)
        at android.view.View.requestLayout(View.java:23093)
        ...
```

```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
    characteristic.value = "hi".toByteArray()
    gatt.writeCharacteristic(characteristic)
  } else {
    textView.text = "Discover error"
  }
}
```

```
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
    characteristic.value = "hi".toByteArray()
    gatt.writeCharacteristic(characteristic)
  } else {
    runOnUiThread {
      textView.text = "Discover error"
    }
  }
}
```

```kotlin
override fun onServicesDiscovered(
  gatt: BluetoothGatt, status: Int
) {
  if (status == GATT_SUCCESS) {
    val characteristic = gatt
        .getService(serviceUuid)!!
        .getCharacteristic(characteristicUuid)!!
    characteristic.value = "hi".toByteArray()
    gatt.writeCharacteristic(characteristic)
  } else {
    runOnUiThread {
      textView.text = "Discover error"
    }
  }
}
```

```
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {
    if (newState == STATE_CONNECTED &&
        status == GATT_SUCCESS
    ) {
      gatt.discoverServices()
    } else {
      runOnUiThread {
        textView.text = "Connect error"
      }
    }
  }

  override fun onServicesDiscovered(
    gatt: BluetoothGatt, status: Int
  ) {
    if (status == GATT_SUCCESS) {
      val characteristic = gatt
          .getService(serviceUuid)!!
          .getCharacteristic(characteristicUuid)!!
      characteristic.value = "hi".toByteArray()
      gatt.writeCharacteristic(characteristic)
    } else {
      runOnUiThread {
        textView.text = "Discover error"
      }
    }
  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

```
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {
    if (newState == STATE_CONNECTED &&
        status == GATT_SUCCESS
    ) {
      gatt.discoverServices()
    } else {
      runOnUiThread {
        textView.text = "Connect error"
      }
    }
  }

  override fun onServicesDiscovered(
    gatt: BluetoothGatt, status: Int
  ) {
    if (status == GATT_SUCCESS) {
      val characteristic = gatt
          .getService(serviceUuid)!!
          .getCharacteristic(characteristicUuid)!!
      characteristic.value = "hi".toByteArray()
      gatt.writeCharacteristic(characteristic)
    } else {
      runOnUiThread {
        textView.text = "Discover error"
      }
    }
  }

  override fun onCharacteristicWrite(
    gatt: BluetoothGatt,
    characteristic: BluetoothGattCharacteristic,
    status: Int
  ) {

  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

```kotlin
override fun onCharacteristicWrite(
  gatt: BluetoothGatt,
  characteristic: BluetoothGattCharacteristic,
  status: Int
) {

}
```

```kotlin
override fun onCharacteristicWrite(
  gatt: BluetoothGatt,
  characteristic: BluetoothGattCharacteristic,
  status: Int
) {
  textView.text = if (status == GATT_SUCCESS)
      "Success" else "Write error"
}
```

```
override fun onCharacteristicWrite(
  gatt: BluetoothGatt,
  characteristic: BluetoothGattCharacteristic,
  status: Int
) {
  textView.text = if (status == GATT_SUCCESS)
      "Success" else "Write error"
}
```

```
W/BluetoothGatt: Unhandled exception in callback
    android.view.ViewRootImpl$CalledFromWrongThreadException:
    Only the original thread that created a view hierarchy can touch its views.
        at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7753)
        at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1225)
        at android.view.View.requestLayout(View.java:23093)
        ...
```

```kotlin
override fun onCharacteristicWrite(
  gatt: BluetoothGatt,
  characteristic: BluetoothGattCharacteristic,
  status: Int
) {
  textView.text = if (status == GATT_SUCCESS)
      "Success" else "Write error"
}
```

```
override fun onCharacteristicWrite(
  gatt: BluetoothGatt,
  characteristic: BluetoothGattCharacteristic,
  status: Int
) {
  runOnUiThread {
    textView.text = if (status == GATT_SUCCESS)
        "Success" else "Write error"
  }
}
```

```kotlin
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {
    if (newState == STATE_CONNECTED &&
        status == GATT_SUCCESS) {
      gatt.discoverServices()
    } else {
      runOnUiThread {
        textView.text = "Connect error"
      }
    }
  }

  override fun onServicesDiscovered(
    gatt: BluetoothGatt, status: Int
  ) {
    if (status == GATT_SUCCESS) {
      val characteristic = gatt
          .getService(serviceUuid)!!
          .getCharacteristic(characteristicUuid)!!
      characteristic.value = "hi".toByteArray()
      gatt.writeCharacteristic(characteristic)
    } else {
      runOnUiThread {
        textView.text = "Discover error"
      }
    }
  }

  override fun onCharacteristicWrite(
    gatt: BluetoothGatt,
    characteristic: BluetoothGattCharacteristic,
    status: Int
  ) {
    runOnUiThread {
      textView.text = if (status == GATT_SUCCESS)
          "Success" else "Write error"
    }
  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

```
val callback = object : BluetoothGattCallback() {
  override fun onConnectionStateChange(
    gatt: BluetoothGatt, status: Int, newState: Int
  ) {
    if (newState == STATE_CONNECTED &&
        status == GATT_SUCCESS) {
      gatt.discoverServices()
    } else {
      runOnUiThread {
        textView.text = "Connect error"
      }
    }
  }

  override fun onServicesDiscovered(
    gatt: BluetoothGatt, status: Int
  ) {
    if (status == GATT_SUCCESS) {
      val characteristic = gatt
          .getService(serviceUuid)!!
          .getCharacteristic(characteristicUuid)!!
      characteristic.value = "hi".toByteArray()
      gatt.writeCharacteristic(characteristic)
    } else {
      runOnUiThread {
        textView.text = "Discover error"
      }
    }
  }

  override fun onCharacteristicWrite(
    gatt: BluetoothGatt,
    characteristic: BluetoothGattCharacteristic,
    status: Int
  ) {
    runOnUiThread {
      textView.text = if (status == GATT_SUCCESS)
          "Success" else "Write error"
    }
  }
}

bluetoothDevice.connectGatt(context, false, callback)
```

Android Bluetooth Low Energy

Android Bluetooth Low Energy

ABLE

Classic Android BLE vs. ABLE

Classic	Able
Mutable	Immutable
Callbacks	Coroutines
Binder threads	Thread-safe
Integer-based states/status	Human readable error and status messages

```
launch {

}
```

```
launch {
    bluetoothDevice.connectGatt(context, false)
}
```

```
launch {
  val result = bluetoothDevice.connectGatt(context, false)
  val gatt = when (result) {
    is Success -> result.gatt
    is Canceled -> TODO()
    is Failure -> TODO()
  }
}
```

```
launch {
  val gatt = bluetoothDevice.connectGattOrThrow(context, false)
}
```

```
launch {
  val gatt = bluetoothDevice.connectGattOrThrow(context, false)
  gatt.discoverServicesOrThrow()
}
```

```
launch {
    val gatt = bluetoothDevice.connectGattOrThrow(context, false)
    gatt.discoverServicesOrThrow()
    gatt.writeCharacteristicOrThrow(
        serviceUuid, characteristicUuid, "hi".toByteArray())
}
```

`writeCharacteristicOrThrow` extension function can be found at: `https://bit.ly/2WUC3ff`

```
launch {
    val gatt = bluetoothDevice.connectGattOrThrow(context, false)
    gatt.discoverServicesOrThrow()
    gatt.writeCharacteristicOrThrow(
        serviceUuid, characteristicUuid, "hi".toByteArray())
}
```

```
launch {
  val gatt = bluetoothDevice.connectGattOrThrow(context, false)
  gatt.discoverServicesOrThrow()
  gatt.writeCharacteristicOrThrow(
      serviceUuid, characteristicUuid, "hi".toByteArray())
  textView.text = "Success"
}
```

```
launch {
    val gatt = bluetoothDevice.connectGattOrThrow(context, false)
    gatt.discoverServicesOrThrow()
    gatt.writeCharacteristicOrThrow(
        serviceUuid, characteristicUuid, "hi".toByteArray())
    textView.text = "Success"
}
```

```
E/AndroidRuntime: FATAL EXCEPTION: DefaultDispatcher-worker-1
    android.view.ViewRootImpl$CalledFromWrongThreadException:
    Only the original thread that created a view hierarchy can touch its views.
        at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7753)
        at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1225)
        at android.view.View.requestLayout(View.java:23093)
        ...
```

```
launch(Dispatchers.Main) {
  val gatt = bluetoothDevice.connectGattOrThrow(context, false)
  gatt.discoverServicesOrThrow()
  gatt.writeCharacteristicOrThrow(
      serviceUuid, characteristicUuid, "hi".toByteArray())
  textView.text = "Success"
}
```

```kotlin
launch(Dispatchers.Main) {
    val gatt = bluetoothDevice.connectGattOrThrow(context, false)
    gatt.discoverServicesOrThrow()
    gatt.writeCharacteristicOrThrow(
        serviceUuid, characteristicUuid, "hi".toByteArray())
    textView.text = "Success"
}
```

```kotlin
launch(Dispatchers.Main) {
  // 1. Connect to peripheral
  val gatt = bluetoothDevice.connectGattOrThrow(context, false)

  // 2. Discover bluetooth services
  gatt.discoverServicesOrThrow()

  // 3. Send "hi" to peripheral
  gatt.writeCharacteristicOrThrow(
      serviceUuid, characteristicUuid, "hi".toByteArray())

  textView.text = "Success"
}
```

```
launch(Dispatchers.Main) {
    val gatt = bluetoothDevice.connectGattOrThrow(context, false)
    gatt.discoverServicesOrThrow()
    gatt.writeCharacteristicOrThrow(
        serviceUuid, characteristicUuid, "hi".toByteArray())
    textView.text = "Success"
}
```

```
launch(Dispatchers.Main) {
  val gatt = withTimeout(10_000L) {
    bluetoothDevice.connectGattOrThrow(context, false)
  }
  gatt.discoverServicesOrThrow()
  gatt.writeCharacteristicOrThrow(
      serviceUuid, characteristicUuid, "hi".toByteArray())
  textView.text = "Success"
}
```

```
launch(Dispatchers.Main) {
  val gatt = withTimeout(10_000L) {
    bluetoothDevice.connectGattOrThrow(context, false)
  }
  gatt.discoverServicesOrThrow()
  gatt.writeCharacteristicOrThrow(
      serviceUuid, characteristicUuid, "hi".toByteArray())
  withContext(Dispatchers.IO) {
    // Write some data to the database.
  }
  textView.text = "Success"
}
```

Able: Android Bluetooth Low Energy

https://github.com/JuulLabs-OSS/able

```
dependencies {
    implementation "com.juul.able:core:0.7.0"
    implementation "com.juul.able:throw:0.7.0"
    implementation "com.juul.able:processor:0.7.0"
    implementation "com.juul.able:timber-logger:0.7.0"
    implementation "com.juul.able:retry:0.7.0"
    implementation "com.juul.able:device:0.7.0"
}
```