eSense Unity Example

Files in this release

```
Unitypackage with all assets and the decoder library:
       eSenseDecoderDemo-EsenseMuscleEEGeniusUnityMobile-2022-3.unitypackage

Android installer packet of the above built demo:
       eSenseDecoderDemo-ESenseMuscleEEGeniusUnityMobile-2022-3.apk
```

Introduction

The example was developed with Unity editor version 2022.3.15f1 (LTS) and depends on several 3rd party assets that are included in the distribution (`eSenseDecoderDemo-EsenseMuscleEEGeniusUnityMobile-2022-3.unitypackage`) in the asset folder. These assets are included in the unitypackage for demonstration purposes only. They are 3rd party assets that have their own licenses, which you can find in the asset folder. You must respect and follow the licenses of the 3rd party assets if you want to use them in your own projects. The decoder library itself (`DecoderClassRelease-v1-0-0–20240113-1.dll`) is built for `<TargetFramework>netstandard2.0</TargetFramework>` and has no dependencies besides that.

Unity folder structure after import of Unity Package:

```
Assets>tree

+---Fonts
¦   +---Courier Prime
¦       +---LICENSE
+---Mindfield
¦   +---Documentation
¦   +---ESenseMuscle EEGeniusDecoder
¦   +---Example
+---Plugins
¦   +---Android
¦   +---BluetoothLEOSX.bundle
¦   ¦   +---Contents
¦   ¦       +---MacOS
¦   ¦       +---_CodeSignature
¦   +---iOS
+---Resources
+---Scenes
+---Shatalmic
    +---Documentation
    +---Editor
    +---Example
      ...
    +---StartingExample
```

The decoder library decodes the eSense ExG (EMG: eSense Muscle or EEG: eSense EEGenius) protocol (time multiplexed EMG or EEG and other data) sent over BLE into different “Packets” and “Reports”:

Decoded Data or Decoder Status	**Data update interval in ms**	**Description**
StatusAndTimingReport	1000	BLE transmission quality measures as: `PacketsCompleteFlag` indicates if every BLE notification was complete. `PacketsPerSecondGoodFlag` indicates if protocol timing tolerances were met.
PacketFuelGauge	1000	Battery status like voltage, current and % battery remaining
PacketAccelerometer	1000	Linear acceleration, rotational acceleration, magnetic field (optional, depends on chipset)
PacketImpedance (only for EEG device)	1000	In EEG devices the electrode impedance is continuously measured
PacketBandpass (device in bandpass mode)	EMG: 10 EEG: 40	EMG: 2 channels with 3 bandpasses in ADC units and RMS uV EEG: 2 channels with 10 bandpasses in ADC units and peak-peak uV
PacketSample (device in sample mode)	EMG: 10 EEG: 12	EMG: 2 channels with 1 sample in ADC units and uV EEG: 2 channels with 3 samples in ADC units and uV. 12ms update interval per 3 samples means 4ms per sample.

1) 3rd party asset: Unity Bluetooth LE Plugin for Android

The decoder library is purely responsible for decoding the ExG protocol. So for using BLE and controlling the eSense device firmware modes any reliable and working BLE library can be used. For demonstration purposes this one has been chosen:

„Unity Bluetooth LE Plugin for Android“ resides in 2 folders:

1. Assets/Plugins
2. Asset/Shatalmic

https://assetstore.unity.com/packages/tools/network/bluetooth-le-for-ios-tvos-and-android-26661

Bluetooth LE for iOS, tvOS and Android by Shatalmic, llc

Shatalmic, llc
www.shatalmic.com
support@shatalmic.com

License agreement: Standard Unity Asset Store EULA
License type: Extension Asset
Latest version: 2.55
Latest release date: Jun 14, 2022
Original Unity version: 2018.4.36 or higher

2) 3rd party asset: Courier Prime Font, monospaced

Courier Prime is a TrueType monospaced font designed specifically for screenplays. It was designed by Alan Dague-Greene for John August and released by Quote-Unquote Apps under the SIL Open Font License (OFL).

Visit http://quoteunquoteapps.com/courierprime for more information and the latest updates.

SIL Open Font License: http://scripts.sil.org/OFL

3) Mindfield Biosystems Ltd. assets:

- `„Asset/Mindfield/ESenseMuscle EEGeniusDecoder/DecoderClassRelease-v1-0-0–20240113-1.dll“` is the compiled decoder class library.
- `„Asset/Mindfield/Example/EsenseExample.cs“` is a Unity C# script based on the asset `„Asset/Shatalmic/Example/StartingExample/StartingExample.cs“.`
  - `„EsenseExample.cs“` demonstrates how to set up the decoder, how to connect to the eSense device and how to control the eSense firmware modes, e.g. selecting ExG data modes and additional data like accelerometer and fuel

    gauge. This is decribed in more detail later is this document.

- `„Asset/Mindfield/Example/EsenseExample.unity“` Unity scene based on the asset `„Asset/Shatalmic/Example/StartingExample/StartingExample.unity“.`
    - Known limitation: Does not scale depending on the aspect ratio. So the last lines of the console panel might be clipped.

Decoder Library Version Changes

`1.0.0`

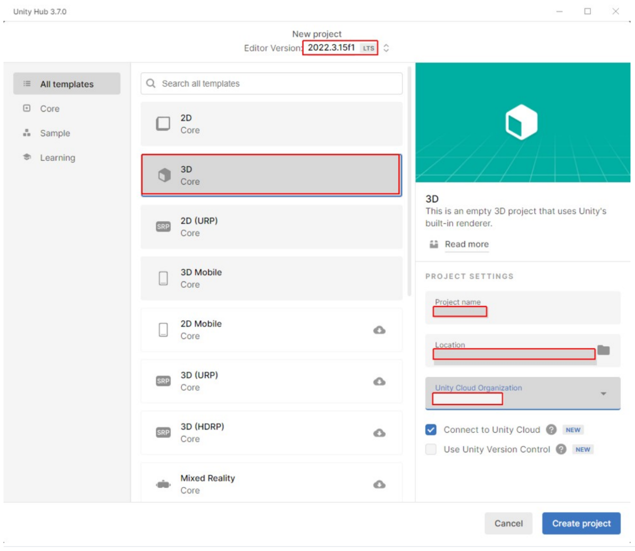
Filename: `DecoderClassRelease-v1-0-0–20240113-1.dll`

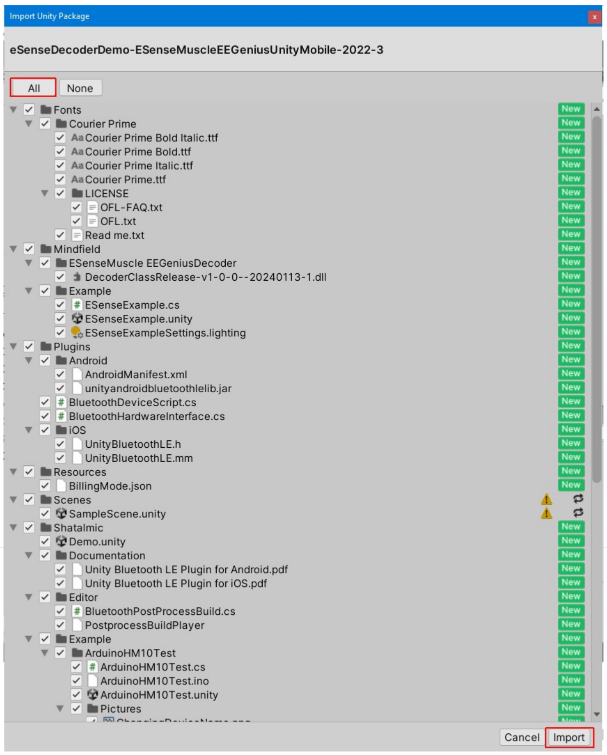
Initial release.

Unity Demo Setup Guide

In Unity Hub create a new project for `Editor version 2022.3.15f1 LTS`. Use the `“3D Core”` template. Fill in your name, location, organization.

In the Unity editor import `„All“` items from the released unitypackage `„eSenseDecoderDemo-EsenseMuscleEEGeniusUnityMobile-2022-3.unitypackage“`




Import Unity Package

eSenseDecoderDemo-ESenseMuscleEEGeniusUnityMobile-2022-3

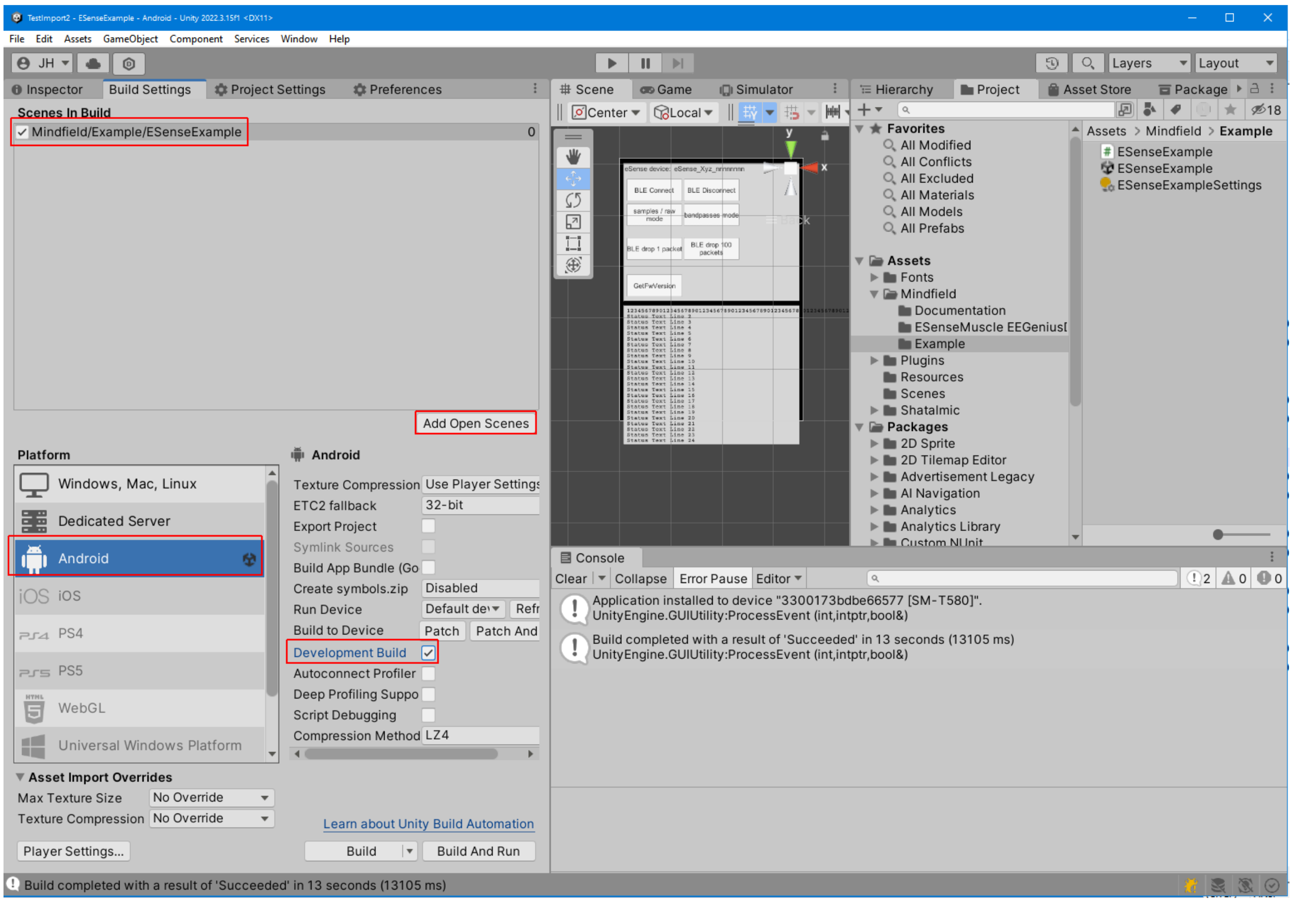
All None

```
▼ ✓ Fonts                                                  New
  ▼ ✓ Courier Prime                                        New
      ✓ Courier Prime Bold Italic.ttf                      New
      ✓ Courier Prime Bold.ttf                             New
      ✓ Courier Prime Italic.ttf                           New
      ✓ Courier Prime.ttf                                  New
    ▼ ✓ LICENSE                                            New
        ✓ OFL-FAQ.txt                                      New
        ✓ OFL.txt                                          New
      ✓ Read me.txt                                        New
▼ ✓ Mindfield                                              New
  ▼ ✓ ESenseMuscle EEGeniusDecoder                         New
      ✓ DecoderClassRelease-v1-0-0--20240113-1.dll         New
  ▼ ✓ Example                                              New
      ✓ ESenseExample.cs                                   New
      ✓ ESenseExample.unity                                New
      ✓ ESenseExampleSettings.lighting                     New
▼ ✓ Plugins                                                New
  ▼ ✓ Android                                              New
      ✓ AndroidManifest.xml                                New
      ✓ unityandroidbluetoothlelib.jar                     New
    ✓ BluetoothDeviceScript.cs                             New
    ✓ BluetoothHardwareInterface.cs                        New
  ▼ ✓ iOS                                                  New
      ✓ UnityBluetoothLE.h                                 New
      ✓ UnityBluetoothLE.mm                                New
▼ ✓ Resources                                              New
    ✓ BillingMode.json                                     New
▼ ✓ Scenes
    ✓ SampleScene.unity
▼ ✓ Shatalmic                                              New
    ✓ Demo.unity                                           New
  ▼ ✓ Documentation                                        New
      ✓ Unity Bluetooth LE Plugin for Android.pdf          New
      ✓ Unity Bluetooth LE Plugin for iOS.pdf              New
  ▼ ✓ Editor                                               New
      ✓ BluetoothPostProcessBuild.cs                       New
      ✓ PostprocessBuildPlayer                             New
  ▼ ✓ Example                                              New
    ▼ ✓ ArduinoHM10Test                                    New
        ✓ ArduinoHM10Test.cs                               New
        ✓ ArduinoHM10Test.ino                              New
        ✓ ArduinoHM10Test.unity                            New
      ▼ ✓ Pictures                                         New
```

Cancel Import

In the Unity editor “Build Settings”

- Open the `“Assets\Mindfield\Example\ESenseExample.unity”` scene.
- `“Add Open Scenes”.`
- Change Platform to `“Android”`.
- Checkmark `“Development Build”` (Important for decoded data debug output to ABD Logcat)



Scenes In Build
Mindfield/Example/ESenseExample 0

Android

Texture Compression	Use Player Settings
ETC2 fallback	32-bit
Export Project	☐
Symlink Sources	☐
Build App Bundle (Go	☐
Create symbols.zip	Disabled
Run Device	Default de
Build to Device	Patch / Patch And
Development Build	✓
Autoconnect Profiler	☐
Deep Profiling Suppo	☐
Script Debugging	☐
Compression Method	LZ4

Asset Import Overrides

Max Texture Size	No Override
Texture Compression	No Override

```
Application installed to device "3300173bdbe66577 [SM-T580]".
UnityEngine.GUIUtility:ProcessEvent (int,intptr,bool&)
Build completed with a result of 'Succeeded' in 13 seconds (13105 ms)
UnityEngine.GUIUtility:ProcessEvent (int,intptr,bool&)
```

Build completed with a result of 'Succeeded' in 13 seconds (13105 ms)

Example Scene and Code

`“EsenseExample.cs”` is based on the 3rd party Unity BLE asset by Shatalmic that is documented in `“Assets/Shatalmic/Documentation/Unity Bluetooth LE Plugin for Android.pdf”.`

`“EsenseExample.cs”` is attached to the scenes `Main Camera`.

The UI Buttons `On Click()` event handlers are set to the according `OnButton...` functions.

`Canvas->Panel->StatusPanel->StatusText` shows some selected info about

- User interactions and
- BLE connection status
- Decoder status

The full info about decoded packet content from this Unity demo currently can only be seen with ADB Logcat (e.g. in Android Studio). However the user can easily extend the demo code and use any decoded packet content to control Unity Objects.

Recommended Logcat filter expressions are:

`package:com.Mindfield.ESenseMuscleEEGeniusUnityMobile`

and optionally:

```
message:PacketSample
message:PacketBandpass
message:PacketFuelGauge
message:PacketAccelerometer
message:PacketImpedance
message:StatusAndTimingReport
```

ESenseExample.cs

```
Line 13, using Mindfield.Esense.BlePacketDecoder;
```

The namespace of the decoder class library

```
Line 237, void InitDecoderInstance()
```

Shows how to set up the `DecoderClass` instance and the required callbacks for further processing of the decoded data. After setting up the callbacks of the instance it is required to make them active by decoder.`RegisterCallbacks()`.

```
Line 336, void Update()
```

Shows the main state machine, that handles BLE scanning, discovering services and characteristics.

Make sure to attempt discovering both the **preferred** and the **compatibility** `ExgServiceUUID` and `ExgMeasurementCharacteristicUUID`.
Only one will be available. The current Firmware uses `ExgFallbackServiceUUID` and the associated characteristics. Future firmware versions however will use `ExgServiceUUID` and the associated characteristics.

```
line 29,
   /// <summary>
   /// Preferred Exg service.
   /// </summary>
   private string ExgServiceUUID = "5e6da50a-7fcc-11eb-9439-0242ac130002";
   private string ExgMeasurementCharacteristicUUID = "5e6da762-7fcc-11eb-9439-0242ac130002";

   /// <summary>
   /// Exg over HR service for compatibility.
   /// </summary>
   private string ExgFallbackServiceUUID = "180d";
   private string ExgFallbackMeasurementCharacteristicUUID = "2a37";
```

The binary data of the notifications received after subscribing `ExgServiceUUID, ExgMeasurementCharacteristicUUID` or `ExgFallbackServiceUUID, ExgFallbackMeasurementCharacteristicUUID` are further processed in line 707 by the notification handler `ProcessExgData(...).`

In line 714 the decoder is called:

`decoder.DecodeBlePacket(Time.unscaledTimeAsDouble, oneBlePacket);`
One or several calls are necessary to complete the decoding, depending on what multiplexed packet types are selected in the firmware.

If the eSense ExG data packet is completely decoded, the according callback with the decoded data will be called.

The ExG protocol modes (bandpass and sample mode) can be mutually exclusively selected. Optional additional protocol modes in arbitrary combination for fuel gauge, accelerometer and impedance can be switched ON and OFF. This is achieved via the CmdService by sending (line 744, `SendCmdString(string cmd)`) and receiving ASCII strings (line 758, `ProcessCmdRead(byte[] bytesRead)`).

```
Line 24,
…
    private string CmdServiceUUID = "6e400001-b5a3-f393-e0a9-e50e24dcca9e";
    private string CmdWriteCharacteristicUUID = "6e400002-b5a3-f393-e0a9-e50e24dcca9e";
    private string CmdReadCharacteristicUUID = "6e400003-b5a3-f393-e0a9-e50e24dcca9e";
```

After subscribing the required BLE characteristics the main state machine handles `States.StartFuelGauge` and `States.StartAccelerometer`.

Commands for controlling the firmware mode via `SendCmdString(string cmd)`:

Command string	**Explanation**
SET_PACKET(<packet_id>,<ON|OFF>)	packet_id is a 0x prefixed hex number.
SET_PACKET(0x25,ON)	ExG mode: eSenseMuscle EMG bandpass mode
SET_PACKET(0x45,ON)	ExG mode: eSenseEEGenius EEG bandpass mode
SET_PACKET(0x20,ON)	ExG mode: eSenseMuscle EMG samples/raw mode
SET_PACKET(0x40,ON)	ExG mode: eSenseEEGenius EEG samples/raw mode
SET_PACKET(0x65,ON)	Optional mode: Fuel Gauge (OFF after FW boot)
SET_PACKET(0x60,ON)	Optional mode: Accelerometer (OFF after FW boot)
SET_PACKET(0x50,ON)	Optional mode (EEG only): Impedance (ON after FW boot)

Each issued command string is answered by the firmware:

```
    /// <summary>
    /// Firmware answer on receiving a command string.
    /// The answer string consists of 2 or 3 lines with LF termination.
    ///  1) echo of the received command
    ///  2) error code: "0" for OK, "1" for error
    ///  3) optional data: e.g. FW version string or data/time etc.
    /// </summary>
    /// <param name="bytesRead"></param>
    private void ProcessCmdRead(byte[] bytesRead)
```

API Reference

Mindfield.Esense.BlePacketDecoder Class Reference

File: DataTypesCallbacks.cs

```
public delegate void StatusAndTimingReportCallbackType(DataTimeIndex timeIndex,
StatusAndTimingReport statusAndTimingReport);

public delegate void DataExgBpCallbackType(DataTimeIndex timeIndex, List<DataExgBandPass> ch1BpList,
List<DataExgBandPass> ch2BpList, List<string> descriptionlist);

public delegate void DataExgSampleCallbackType(List<DataTimeIndex> timeIndexList,
List<DataExgSample> ch1SampleList, List<DataExgSample> ch2SampleList, List<string> descriptionlist);

public delegate void DataFuelGaugeCallbackType(DataTimeIndex timeIndex, DataFuelGauge fuelGauge);

public delegate void DataImpedanceCallbackType(DataTimeIndex timeIndex, DataImpedance impedance);

public delegate void DataAccelerometerCallbackType(DataTimeIndex timeIndex, DataAccelerometer
accelerometer);
```

DataTimeIndex Class Reference

Timing information for ExG samples. More...

Public Member Functions

DataTimeIndex ()

Initializes a new instance of the DataTimeIndex class.

DataTimeIndex (**DataTimeIndex** element)

Initializes a new instance of the DataTimeIndex class.

void **ResetDataTimeIndex** ()

Resets the DataTimeIndex.

Properties

int **SampleIndex** `[get, set]`

Gets or sets the unique index of a sample. Normally increments by 1 with each sample.

double **NominalTimeSec** `[get, set]`

Gets or sets an idealized time index for a sample, calculated from the SampleIndex and the samplerate.

double **RealTimeSec** `[get, set]`

Gets or sets the real time index (from the host system clock) of a sample. This may have lots of timing jitter because the BLE stack decides about the transmission timing.

StatusAndTimingReport Class Reference

Properties

int **LastExgPacketId** `[get, set]`

string **LastExgPacketIdAsText** `[get, set]`

bool **PacketsCompleteFlag** `[get, set]`

int **PacketsDroppedTotal** `[get, set]`

float **PacketsPerSecond** `[get, set]`

float **PacketsPerSecondAverage** `[get, set]`

bool **PacketsPerSecondGoodFlag** `[get, set]`

DataExgBandPass Class Reference

Properties

int **ChannelAdcUnits**`[get, set]`

Gets or sets the value of a channel in ADC units.

double **ChannelMicroVolts**`[get, set]`

Gets or sets the value of a channel in micro volts units.

DataExgSample Class Reference

Properties

int **ChannelAdcUnits**`[get, set]`

Gets or sets the value of a channel in ADC units.

double **ChannelMicroVolts**`[get, set]`

Gets or sets the value of a channel in micro volts units.

DataFuelGauge Class Reference

The device Li-Ion battery is monitored by a fuel gauge chip. Charge counting values are ony accurate after initially fully charging the device. More...

Properties

int **Voltage**`[get, set]`

Gets or sets the battery voltage in mV DC.

int **Current**`[get, set]`

Gets or sets the battery current in mA DC.

int **CoulombCounter**`[get, set]`

Gets or sets the battery charge in mAh.

float **PercentageRemaining**`[get, set]`

Gets or sets the battery remaining charge in percentage.

int **ConversionCounter**`[get, set]`

Gets or sets the battery fuel gauge chip ADC conversion counter. Increments by 1 with each conversion.

int **Temperature**`[get, set]`

Gets or sets the battery temperature in 1/10 °C.

Detailed Description

The device Li-Ion battery is monitored by a fuel gauge chip. Charge counting values are ony accurate after initially fully charging the device.

DataImpedance Class Reference

Properties

float **Ch1Abs**`[get, set]`

float **Ch1Phi**`[get, set]`

float **Ch2Abs**`[get, set]`

float **Ch2Phi**`[get, set]`

DataAccelerometer Class Reference

Properties

float **LinearX**`[get, set]`

float **LinearY**`[get, set]`

float **LinearZ**`[get, set]`

float **GyroX**`[get, set]`

float **GyroY**`[get, set]`

float **GyroZ**`[get, set]`

float **MagneticX**`[get, set]`

float **MagneticY**`[get, set]`

float **MagneticZ**`[get, set]`