Phantom SDK v1.0

The Erbessd Instruments Phantom SDK allows to interact with the Phantom sensor network.

The SDK is written in C# and is supported both in Windows and Linux (using mono).

Phantom SDK Basics

The Phantom SDK is message based, to initialize you start the network listening threads calling:

```
public void start(ProcessPhantomMessages messageProcessor, bool receive_vibration_data)
```

This function takes two parameters, the first will be a delegate that will handle all Phantom incoming messages.

The second parameter enables or disables receiving of long vibration data. The Phantom gateway sends two types of messages, short using the UDP protocol and long using the TCP protocol. You can receive the short messages from multiple computers in the network, but only one can receive the long TCP messages. This parameter controls where or not long TCP messages will be received.

The message processor delegate will look like this:

```
void processMessages(Phantom.Messages message, Phantom.MessageData data)
```

This delegate will be called every time a new phantom message arrives. The message parameter will tell the type of message and the second parameter will have message specific data (in the case of vibration measurements it will have the array of the signal measured).

Note that this function will be called from the context of different threads, if you want to update UI or serialize the messages you need a mechanism to bring this all back into the main thread (Invoke or a Queue).

To stop the PhantomLib threads call:

```
public void stop()
```

Phantom Messages

Phantom.Messages.RECEIVED_PHANTOM_ACCEL_DATA

This message contains the long measurement data from the Phantom acceleration node.

```
        public class MessageDataEIMONanoAccel : MessageData
        {
            public Reason recording_reason;
            public string phantom_code;
            public float temperature;
            public short[] channel1;
            public short[] channel2;
            public short[] channel3;
            public double calibration;
            public int sample_rate;
        }
```

The recording_reason can be any of

```
        public enum Reason
        {
            REQUESTED = 1,
            SCHEDULED = 2,
            ALARM = 3
        }
```

phantom_code is the phantom serial number.

temperature will be the node internal radio transmitter temperature in C.

channel1, channel2 and channel3 will be an array of signed 16bit integers that contain the vibration data, channel3 can be null in the case of the biaxial acceleration node.

calibration is the calibration of the data. Multiplying the array members for this number will result in the value of the vibration in G (9.8m/s^2).

sample_rate has the sample rate of the measurement.

Phantom.Messages.RECEIVED_PHANTOM_TEMP_DATA

This message contains the data from the Phantom temperature nodes, either thermocouple or infrared.

```
        public class MessageDataEIMONanoTemp : MessageData
        {
            public Reason recording_reason;
            public string phantom_code;
            public TemperatureType type;
            public float temperature;
            public float battery;
            public float ambient_temperature;
            public float module_temperature;
        }
```

The recording_reason can be any of

```
        public enum Reason
        {
            REQUESTED = 1,
            SCHEDULED = 2,
            ALARM = 3
        }
```

phantom_code is the phantom serial number.

type can be

```
        public enum TemperatureType
        {
            THERMOCOUPLE = 1,
            INFRARED = 2
        }
```

module_temperature will be the node internal radio transmitter temperature in C.

battery the current battery voltage.

ambient_temperature is the ambient temperature in celsius as measured by the infrared module.

temperature is the “object temperature” in C for the infrared module or the thermocouple temperature for the thermocouple module.

Phantom.Messages.RECEIVED_PHANTOM_ACCEL_SETTINGS

This message contains the current settings of the Phantom acceleration node.

```
        public class MessageEIMONanoAccelSettings : MessageData
        {
            public string phantom_code;
            public int send_interval;
            public int sample_rate;
            public int samples_to_get;
            public int range;
            public float alarm1;
            public float alarm2;
            public float alarm3;
            public int alarmcheck_interval;
        }
```

phantom_code is the phantom serial number.

send_interval is the send interval of full acceleration data in minutes.

sample_rate is the sample rate of the acceleration node.

samples_to_get is the number of samples that will be recorded.

range is the range in Gs of the sensor.

alarm1/alarm2/alarm3 are the current alarm configurations in mm/s of the node.

alarmcheck_interval is how often would the alarms will be check, in seconds.

Phantom.Messages.RECEIVED_PHANTOM_ACCEL_STATE

This message contains the short state of the Phantom acceleration node.

```
        public class MessageEIMONanoAccelState : MessageData
        {
            public string phantom_code;
            public float battery;
            public float temperature;
            public float rms1;
            public float rms2;
            public float rms3;
        }
```

phantom_code is the phantom serial number.

battery the current battery voltage in V.

temperature will be the node internal radio transmitter temperature in C.

rms1, rms2, rms3 are the speed rms as measured by the sensor, in mm/s.

Phantom.Messages.RECEIVED_PHANTOM_CURRENT_DATA

This message contains the data of the Phantom current node.

```
        public class MessageEIMONanoCurrentData : MessageData
        {
            public string phantom_code;
            public float battery;
            public float module_temperature;
            public float current1;
            public float current2;
            public float current3;
        }
```

phantom_code is the phantom serial number.

battery the current battery voltage in V.

module_temperature will be the node internal radio transmitter temperature in C.

current1/current2/current3 are the current measurements in A.

Phantom.Messages.RECEIVED_PHANTOM_RPM_DATA

This message contains the data of the Phantom RPM node.

```
        public class MessageEIMONanoRPMData : MessageData
        {
            public string phantom_code;
            public float battery;
            public float module_temperature;
            public float rpm;
        }
```

phantom_code is the phantom serial number.

battery the current battery voltage in V.

module_temperature will be the node internal radio transmitter temperature in C.

rpm is the RPM measurement.

Modifying configuration settings

```
        public void send_einano_accel_config(string ip_address, string phantom_code, int
send_interval, float alarm1, float alarm2, float alarm3, int alarmcheck_interval)
```

This function can be used to modify the Phantom acceleration node settings.

ip_address is the ip address of the gateway the node is connected to (you need to keep track of it by listening to the ID messages)

phantom_code is the serial number of the phantom acceleration node.

send_interval is the send interval of the long acceleration measurement in minutes. Modifying this value will affect battery life.

alarm1, alarm2 and alarm3 are the alarm configuration in mm/s.

alarmcheck_interval is interval in which the alarm will be checked, in seconds. Modifying this value will affect battery life.

The configuration request update will be sent to the gateway which will store it and pass it into the node the next time it connects to request its configuration. It will take a while for it to become active.