Moodmetric Software Development Kit

2015-11-08

Version 1.4

MOODMETRIC

MASTER YOUR MIND

Contents

What is Moodmetric

Moodmetric ring is a powerful and easy-to-use tool for detecting emotional intensity of the user. Moodmetric measures electrodermal activity (EDA) to do this. Information about the operating principle of the Moodmetric ring and the science behind it can be found at:

http://www.moodmetric.com/moodtech-science-behind-moodmetric/

The signal quality of Moodmetric has been verified by Scientists of the Finnish Institute of Occupational Health:

J. Torniainen, B. Cowley, A. Henelius, K. Lukander, and S. Pakarinen, "Feasibility of an electrodermal activity ring prototype as a research tool," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pp.6433-6436, 25-29 Aug. 2015

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7319865

Moodmetric can provide a real Wow factor into your app.

License

To apply Moodmetric technology in your application, you must apply for a license from Moodmetric. Failure to use the Moodmetric ring interface in any other way than presented in this document will result in discontinuation of the license contract.

Disclaimer

Moodmetric's products are not medical devices and should not be used to diagnose or treat any medical conditions. Moodmetric's products are not authorized for use as critical components in life support devices or systems. Moodmetric does not guarantee the wellbeing of the user.

Purpose of this document

This document allows you to develop your own application, which uses emotion data provided by the Moodmetric ring. To ensure operation of 3rd party applications in any mobile platform or computer operating system, Moodmetric provides interface documentation directly to the ring. The Moodmetric ring uses Bluetooth Smart, so the platform running the 3rd party application must be Bluetooth Smart Ready.

In addition to interface specification, this document includes example codes for making the connection and reading data for following platforms:

- IOS
- Android

What you get

The Moodmetric ring provides instantaneous and longer term indicators of the emotional intensity of the user. Also the acceleration of the ring is available. In the following, the different indicators are listed:

- Instant indicator
- MM number
- Acceleration
- Raw data

You will have to experiment to see which of these suit your application best. In the following the indicators are explained in more detail.

Instant indicator

Instant indicator is a fast indicator how the person reacts to different stimuli. These can be what you see, hear, smell or sense by touch at a certain moment. The response can be seen as an upward spike, which starts approximately after 1.5 seconds after the stimulus and peaks approximately after 2.5-3.0 seconds. Return to base level takes a couple of seconds. The 1.5 second latency is usually very constant among the population. Constant latency is very useful in many applications. The amplitude of the response is typically between 0…800 units above the base level. A very strong response can be as high as 1600. The base level is set to 2^{14}=16384.

Moodmetric cannot specify these times or amplitudes precisely because we are dealing with human beings. You have to find suitable parameters for your application by experimenting. See typical EDA waveform in Fig. 1.

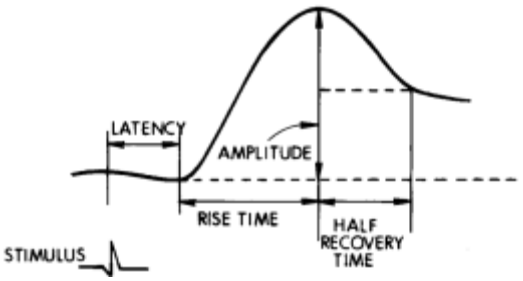

Fig. 1 EDA waveform

When the Moodmetric ring is put in finger, the instant indicator takes 5 seconds to stabilize. If the ring is in finger when connection is made, the instant indicator is ready instantly. If the ring is taken briefly off the finger while still connected, the instant indicator goes to its base level until a good skin contact is regained.

MM number

The Moodmetric number or the MM number is a continuous indicator how strongly you have felt during the last minutes. MM number is an intuitive way to tell what the user's emotional intensity is right now. Low MM numbers can be achieved at a calm and quiet environment. High MM numbers are typical when angry, stressed or when in intensive social situations. The MM number is always between 0 and 100, and moves slowly. MM number is comparable between different users. More information about the behavior of the MM number can be found at:

http://www.moodmetric.com/emotional-intensity/

When the Moodmetric ring is put in finger, the MM number takes some time to stabilize. If the ring is in finger when connection is made, the MM number is ready instantly. If the ring is taken briefly off the finger while still connected, the MM number freezes until a good skin contact is regained.

Acceleration

Data of 3-axis accelerometer inside the ring is provided. It might be useful in interface applications and such.

Notifications

You will also get the following notifications:

- MM number notification
- Relax number notification
- Strong reaction notification
- Some technical notifications

The MM number notification tells if the person is very stressed or emotionally loaded. High MM notification is issued when MM number is elevated for over 5 minutes.

The Relax notification tells if the person has successfully managed to relax for example in a meditation exercise. Relax notification is issued when MM number is low for over 5 minutes. This feature requires a ring with application version of 1.0.1 or above.

The strong reaction notification tells if a strong emotional reaction has been detected at a certain moment. You can test responses to images, sounds and such things.

Interface specification

The interface enables continuous data streaming and notifications. The Moodmetric service contains both options.

The UUID of the Moodmetric service is

`dd499b70-e4cd-4988-a923-a7aab7283f8e`

It contains two characteristics

- Streaming
- Notifications
- Raw data

Battery level and Device information can be read on services `0x180f`, `0x1800` and `0x180a` as specified at Bluetooth.org:

https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

Interfacing any of the other characteristics of the ring is not allowed by a 3rd party application. Failure to comply will result in termination of the license contract.

Streaming characteristic

The streaming characteristic is specified in the following. It provides the streaming data.

The UUID of the streaming characteristic is

`a0956420-9bd2-11e4-bd06-0800200c9a66`

It supports the following operations:

- Read
- Notify

If you wish to poll, use read. If you wish to receive packets at continuous intervals, enable notifications on this characteristic. The packet interval is 3 packets per second by default when notifications are used. Default sample rate of 3 S/s has been optimized for lowest possible current consumption without compromising the EDA waveform. Same update rate is used for each of the indicators, including the accelerometer.

Length of the payload is 7 bytes. The contents of the payload are specified in Table 1.

Table 1. Packet payload content

Payload bytes						
0	1	2	3	4	5	6
Status bits	MM	Instant		a_x	a_y	a_z

Bytes	Description	Range (dec)	Notes
[0]	Status bits	-	See status bits in Table 2
[1]	MM number	0…100	Time constant is in the minute range
[2…3]	Instant	0…65535	Base level is 16384
[4]	a_x	0…255	Acceleration X, 0 = -2 g, 127 = 0 g, 255 = +2 g.
[5]	a_y	0…255	Acceleration Y, 0 = -2 g, 127 = 0 g, 255 = +2 g.
[6]	a_z	0…255	Acceleration Z, 0 = -2 g, 127 = 0 g, 255 = +2 g.

Table 2. Status bits in data packet

Status bits	State if 1	Notes
0 (LSB)	MM not ready	After put in finger, or with bad skin contact
1	Ring in finger	Tells if the ring is in finger
2	Battery is out	Ring will stop sending
3	Any reaction	Peak detector for instant indicator
4	Strong reaction	Strong peak detector for instant indicator
5	MM notification	MM elevated for 5 min
6	Relax notification	MM low for 5 min
7 (MSB)	Reserved for future use	Reads as zero

The ‘MM not ready’ bit is up right after the ring is put in finger since it takes some time for the MM get its first value. This bit is also up when bad skin contact is detected. Then the MM number freezes and the Instant indicator is forced slowly to its base level until good contact is regained. Then ‘MM not ready’ bit goes back down.

The sample rate can be changed by writing a single byte to the sample rate characteristic in the Moodmetric service. Reading is also enabled. The sample rate applies for both EDA based signals and for the accelerometer. The ring will save the sample rate value in its non-volatile memory. This feature requires a ring with application version of 1.0.1 or above.

The UUID of the sample rate characteristic is:

`2c1531b0-4bd2-11e5-b970-0800200c9a66`

The value must be set between 1-16 S/s. Increasing sample rate increases power consumption. See Table 3. Furthermore, increasing the sample rate above 4 S/s will disable internal memory of the ring. You will notice this when your Moodflower in the mobile App doesn't load and memory consumption always shows 0.0%.

Table 3. Power consumption with different sample rates

Sample rate setting (S/s)	Relative power consumption (%)
3	0
4	+30
8	+116
16	+245

Notification characteristic

The notification characteristic is specified in the following. It provides the notification data.

The UUID of the notification characteristic is

`c48650d0-a2d8-11e4-bcd8-0800200c9a66`

It supports the following operations:

- Read
- Notify

If you wish to poll, use read. If you wish to receive packets only when the notification flags appear or disappear, enable notifications on this characteristic. Notification is sent only when the status bits change state. Length of the payload is 1 byte. The contents of the payload are specified in Table 4.

Table 4. Status bits in notification characteristic

Status bits	State if 1	Notes
0 (LSB)	Reserved for future use	Reads as zero
1	Ring in finger	Tells if the ring is in finger
2	Battery is out	Ring will stop sending
3	Reserved for future use	Reads as zero
4	Strong reaction	Strong peak detector for instant indicator
5	MM warning	MM elevated for 5 min
6	Relax notification	MM low for 5 min
7 (MSB)	Reserved for future use	Reads as zero

Raw data characteristic

The Raw data characteristic is specified in the following. It provides the raw skin resistance.

The UUID of the streaming characteristic is

`f1b41cde-dbf5-4acf-8679-ecb8b4dca6ff`

It supports the following operations:

- Read
- Notify

If you wish to poll, use read. If you wish to receive packets at continuous intervals, enable notifications on this characteristic. Length of the payload is 2 bytes. This feature requires a ring with application version of 1.0.1 or above.

The raw skin resistance R_{skin} is provided without any smoothing or artefact rejection. Thus, this characteristic must be used with discretion. 0 means zero Ohms and 65535 means 16 MOhms. To get conductance in Siemens, take $1/R_{skin}$. Beware of zero-division. To get Microsiemens, multiply with 1 000 000.

Advertising

The Moodmetric ring advertises for one minute at intervals of ~0.15 seconds after put in finger and after connection is dropped. After one minute in finger, advertisement interval slows down to ≥1.0 seconds to preserve battery. After one minute out of finger, advertisement stops. The Moodmetric ring is unidirectional connectable. Advertisement packet contains the Moodmetric service UUID.

Background execution on IOS

On IOS, background execution is limited. In general, you have to have your application started by the user. After that, the application can receive and process data from the ring in the foreground or in the background. If the user kills the app or the app hasn't been started since last reboot, no data can be received from the ring.

For detailed information about background execution on IOS, visit:

https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html

Example code for IOS

The following sections show example Swift code for iOS, how to make a connection to the ring and how to read data from it. The code is adapted from the iOS demo application included in the SDK.

Finding a ring

```
override func viewDidLoad() {
    super.viewDidLoad()
    centralManager = CBCentralManager(delegate: self, queue: nil)
    log("waiting for bluetooth to become available")
}

func centralManagerDidUpdateState(central: CBCentralManager!) {
    if central.state == .PoweredOn {
        log("bluetooth enabled, scanning for rings")
        // Start scanning for devices that support the Moodmetric service
        centralManager.scanForPeripheralsWithServices([mmServiceUUID], options: nil)
    }
}

func centralManager(central: CBCentralManager!, didDiscoverPeripheral peripheral:
CBPeripheral!, advertisementData: [NSObject : AnyObject]!, RSSI: NSNumber!) {
   log("found a ring, connecting to it")
   // Stop scanning
   centralManager.stopScan()
   // Connect to the ring
   connect(peripheral)
}

let mmServiceUUID = CBUUID(string:"dd499b70-e4cd-4988-a923-a7aab7283f8e")
```

Connecting to the ring and enabling notifications

```
func connect(peripheral: CBPeripheral!) {
   centralManager.connectPeripheral(peripheral, options: nil)
   // We need to have a reference to the peripheral because the connection
   // attempt will be canceled when the peripheral is deallocated
   self.peripheral = peripheral
}

func centralManager(central: CBCentralManager!, didConnectPeripheral peripheral:
CBPeripheral!) {
    log("connected, discovering services")
    peripheral.delegate = self
    peripheral.discoverServices([mmServiceUUID])
}

func peripheral(peripheral: CBPeripheral!, didDiscoverServices error: NSError!) {
    // Loop discovered services to find the MM service
    // (though there should be only one)
    for s in peripheral.services {
        if let service = s as? CBService {
            if service.UUID == mmServiceUUID {
                log("discovered MM service")
                // Once we have the service we still need to discover
                // its characteristics
                peripheral.discoverCharacteristics([streamingCharacteristicUUID],
                                                   forService: service)
                return
            }
        }
    }
}

func peripheral(peripheral: CBPeripheral!, didDiscoverCharacteristicsForService
service: CBService!, error: NSError!) {
    // Loop discovered characteristics to find the streaming characteristic
    for ch in service.characteristics {
        if let characteristic = ch as? CBCharacteristic {
            if characteristic.UUID == streamingCharacteristicUUID {
                log("discovered streaming characteristic, enabling notifications")
                // To receive measurements from the ring we
                // need to enable notifications
                peripheral.setNotifyValue(true, forCharacteristic: characteristic)
                return
            }
        }
    }
}

let streamingCharacteristicUUID = CBUUID(string:"a0956420-9bd2-11e4-bd06-0800200c9a66")
```

Receiving and decoding data

```
func peripheral(peripheral: CBPeripheral!, didUpdateValueForCharacteristic
characteristic: CBCharacteristic!, error: NSError!) {
    let data = characteristic.value
    var payload = [UInt8](count: data.length, repeatedValue: 0)
    data.getBytes(&payload, length:data.length)
    // Decode payload
    let status = Int(payload[0])
    let mm = Int(payload[1])
    // Instant EDA is in payload bytes 2 and 3 in big-endian format
    let instant = (Int(payload[2]) << 8) | Int(payload[3])
    let ax = Double(payload[4])/255*4 - 2
    let ay = Double(payload[5])/255*4 - 2
    let az = Double(payload[6])/255*4 - 2
    // Acceleration magnitude (g)
    let a = sqrt(ax*ax + ay*ay + az*az)
    log(String(format: "st:%02x mm:%d eda:%d a:%.2f",
               status, mm, instant, a))
}
```

Example code for Android

The following sections show example code for Android. The code is adapted from the Android demo application included in the SDK.

Finding a ring

```
private void findRing() {
    // Get the bluetooth manager from Android
    final BluetoothManager bluetoothManager =
        (BluetoothManager)getSystemService(Context.BLUETOOTH_SERVICE);
    if (bluetoothManager == null) {
        log("bluetooth service not available");
        return;
    }
    // Get a reference to the bluetooth adapter of the device
    bluetoothAdapter = bluetoothManager.getAdapter();
    if (bluetoothAdapter == null) {
        log("bluetooth not supported");
        return;
    }
    // Check that bluetooth is enabled.
     if (!bluetoothAdapter.isEnabled()) {
        log("bluetooth not enabled");
        return;
    }
    // Start scanning for devices that advertise mmServiceUUID
    bluetoothAdapter.startLeScan(
        new UUID[] { mmServiceUUID },
        new BluetoothAdapter.LeScanCallback() {
            @Override
            public void onLeScan(final BluetoothDevice device,
                int rssi, byte[] scanRecord) {
                log("found ring %s, rssi: %d", device.getAddress(), rssi);
            }

        });
}

private static final UUID mmServiceUUID =
    UUID.fromString("dd499b70-e4cd-4988-a923-a7aab7283f8e");
```

Connecting to the ring and enabling notifications

```
private void connectToRing(BluetoothDevice device) {
    device.connectGatt(DemoActivity.this, false, gattCallback)
}

private final BluetoothGattCallback gattCallback =
    new BluetoothGattCallback() {
        @Override
        public void onConnectionStateChange(final BluetoothGatt gatt,
            int status, int newState) {
            if (newState == BluetoothProfile.STATE_CONNECTED) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        log("connection established, discovering services");
                        // Once we have connection start BLE service discovery
                        gatt.discoverServices();
                    }
                });
            }
        }

        @Override
        public void onServicesDiscovered(final BluetoothGatt gatt, int status) {
            runOnUiThread(
                    new Runnable() {
                        @Override
                        public void run() {
                            enableStreamingNotification(gatt);
                        }
                    });
        }

        private void enableStreamingNotification(BluetoothGatt gatt) {
            log("services discovered, enabling notifications");
            // Get the MM service
            BluetoothGattService mmService = gatt.getService(mmServiceUUID);
            // Get the streaming characteristic
            BluetoothGattCharacteristic streamingCharacteristic =
                    mmService.getCharacteristic(streamingCharacteristicUUID);
            // Enable notifications on the streaming characteristic
            gatt.setCharacteristicNotification(streamingCharacteristic, true);
            // For some reason the above does not tell the ring that it should
            // start sending notifications, so we have to do it explicitly
            BluetoothGattDescriptor cccDescriptor =
                streamingCharacteristic.getDescriptor(
                    clientCharacteristicConfigurationUUID);
            cccDescriptor.setValue(
                BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
            gatt.writeDescriptor(cccDescriptor);
        }
    };

private static final UUID streamingCharacteristicUUID =
    UUID.fromString("a0956420-9bd2-11e4-bd06-0800200c9a66");
// Defined by the BLE standard
private static final UUID clientCharacteristicConfigurationUUID =
    UUID.fromString("00002902-0000-1000-8000-00805F9B34FB");
```

Receiving and decoding data

```
@Override
public void onCharacteristicChanged(
    BluetoothGatt gatt, BluetoothGattCharacteristic characteristic) {
    byte[] payload = characteristic.getValue();
    // Decode payload
    int status = payload[0] & 0xff;
    int mm = payload[1] & 0xff;
    // Instant EDA value is in payload bytes 2 and 3 in big-endian format
    int instant = ((payload[2] & 0xff) << 8) | (payload[3] & 0xff);
    // Acceleration in x, y and z directions (unit is g)
    double ax = (payload[4] & 0xff)*4/255.0 - 2;
    double ay = (payload[5] & 0xff)*4/255.0 - 2;
    double az = (payload[6] & 0xff)*4/255.0 - 2;
    // Acceleration magnitude
    double a = Math.sqrt(ax*ax + ay*ay + az*az);
    log("st:%02x\tmm:%d\teda:%d\ta:%.1f", status, mm, instant, a);
}
```

Revision history

1.0	2015-03-01	Initial version
1.1	2015-03-21	Clarified definition of instant indicator and MM number
1.2	2015-08-26	Added adjusting sample rate and notification for successful relaxation
1.3	2015-09-08	Added raw data output
1.4	2015-11-08	Added reference to scientific verification of Moodmetric and remarks about increased sample rate