EE523
Smart Alignment

Kevin Egedy

Smart Alignment

Smart alignment passively improves posture by notifying the user's spine curvature. This product is based on existing uprightpose, however it is composed of common devices available through Arduino.

Smart Alignment

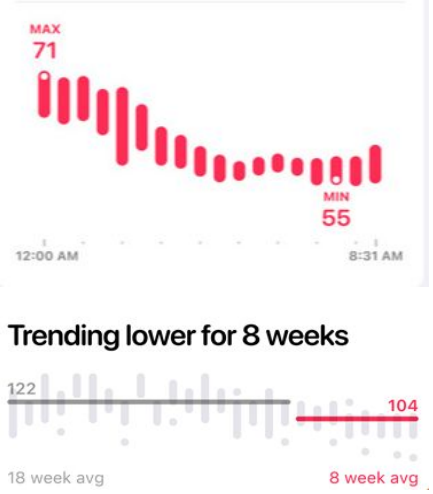
Upright

App Inspirations

Display health information to benefit the user.

- Apple's heart rate display
- Pie chart comparing good vs bad posture

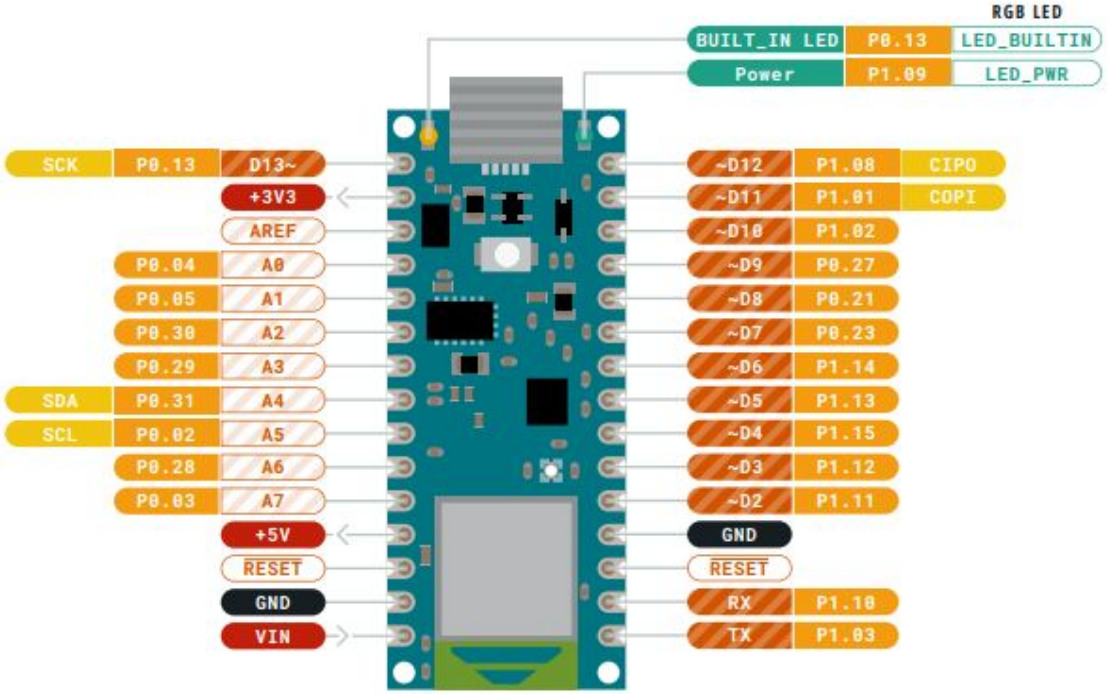
Apple



Upright

Components

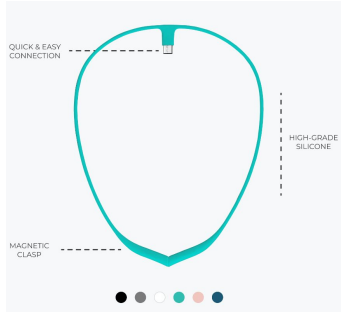
Arduino Nano 33 BLE Pinout



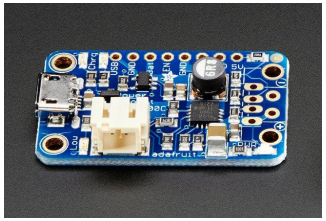
* INPUT-VIN: 4.5-21V

Case

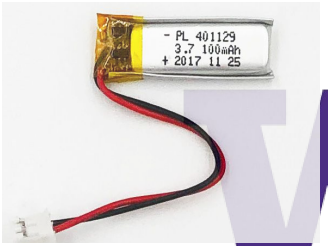
Necklace



5V Lipo Boost

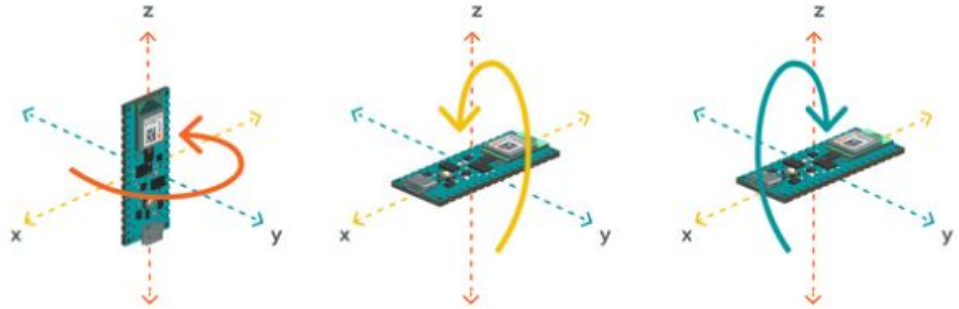


100 mAh Lipo Bat

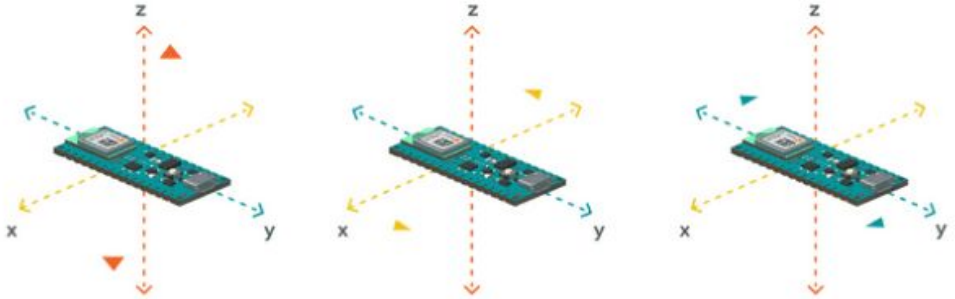


Sensors: 9-axis

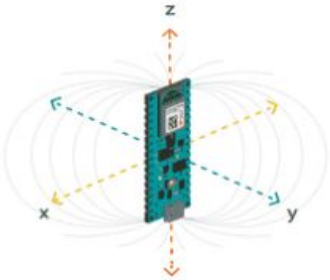
Gyroscope



Accelerometer



Magnetometer

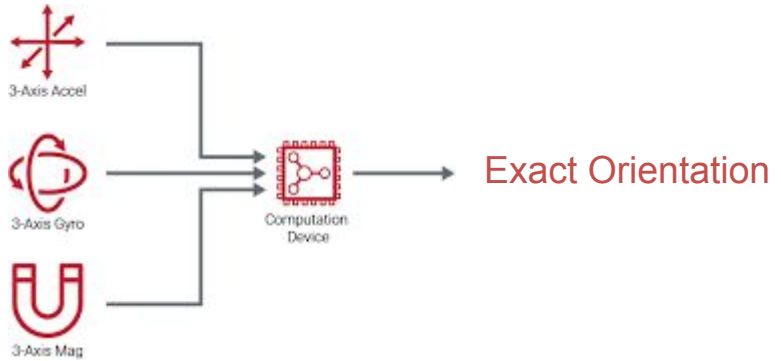


AHRS

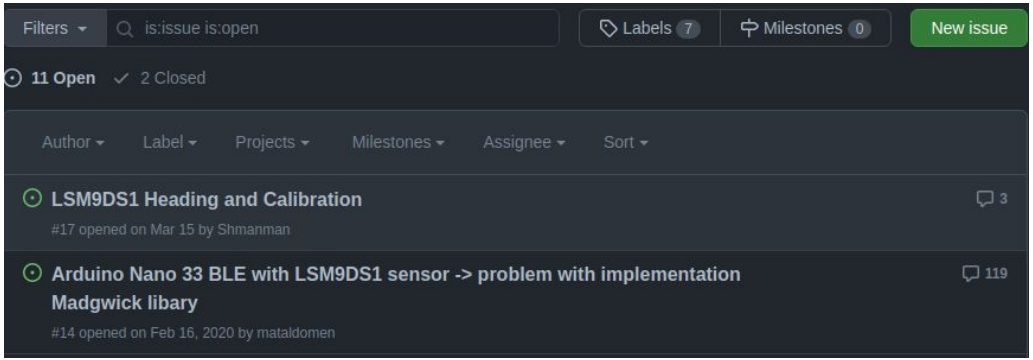
AHRS (Altitude and Heading Reference System)
Determines orientation using

- Mahony,
- Madgwick,
- NXP Fusion,
- etc fusion filters.

Returns pitch, yaw, and roll



https://forum.arduino.cc/t/ble-sense-ahrs/636949
https://github.com/kriswiner/LSM9DS1/issues



Resources

Arduino Nano 33 BLE Pinout

Component	Library
I2C OLED Display Module	Adafruit_SSD1306
Bluetooth Low Energy (BLE)	ArduinoBLE
Arduino Nano IMU	LSM9DS1
Adafruit AHRS	AHRS

Android

https://github.com/android/connectivity-samples

https://punchthrough.com/android-ble-guide/

Demo: Arduino Nano 33 BLE

```
31 // BLE IMU Service
32 BLEService imuService("180F");
33
34 // BLE IMU Level Characteristic
35 BLEUnsignedCharCharacteristic imuLevelChar0("2A19",  // standard 16-bit characteristic UUID
36     BLERead | BLENotify); // remote clients will be able to get notifications if this characteristic changes
37 BLEUnsignedCharCharacteristic imuLevelChar1("2A20", BLERead | BLENotify);
38 BLEUnsignedCharCharacteristic imuLevelChar2("2A21", BLERead | BLENotify);
39
```

```
 99 void setup() {
100     /* Set a local name for the BLE device
101      This name will appear in advertising packets
102      and can be used by remote devices to identify this BLE device
103      The name can be changed but maybe be truncated based on space left in advertisement packet
104   */
105   BLE.setLocalName("SmartAlignment");
106   BLE.setAdvertisedService(imuService); // add the service UUID
107   imuService.addCharacteristic(imuLevelChar0); // add the imu level characteristic
108   imuService.addCharacteristic(imuLevelChar1);
109   imuService.addCharacteristic(imuLevelChar2);
110   BLE.addService(imuService); // Add the imu service
111   imuLevelChar0.writeValue(0); // set initial value for this characteristic
112   imuLevelChar1.writeValue(0);
113   imuLevelChar2.writeValue(0);
114
115   /* Start advertising BLE.  It will start continuously transmitting BLE
116      advertising packets and will be visible to remote BLE central devices
117      until it receives a new connection */
118   // start advertising
119   BLE.advertise();
120   Serial.println("Bluetooth device active, waiting for connections...");
121 }
122
```

Demo: Arduino Nano 33 BLE

```
149 void loop() {
150
151   // wait for a BLE central
152   BLEDevice central = BLE.central();
153
154   // if a central is connected to the peripheral:
155   if (central) {
156     Serial.print("Connected to central: ");
157     // print the central's BT address:
158     Serial.println(central.address());
159     // turn on the LED to indicate the connection:
160     digitalWrite(LED_BUILTIN, HIGH);
161
162     // check the imu level every 500ms
163     // while the central is connected:
164     while (central.connected()) {
165       long currentMillis = millis();
166       // if 500ms have passed, update IMU:
167       if (currentMillis - previousMillis >= 500) {
168         previousMillis = currentMillis;
169         if (IMU.gyroscopeAvailable()) {
170           IMU.readGyroscope(x, y, z);
171           u = int(x);
172           v = int(y);
173           w = int(z);
174           imuLevelChar0.writeValue(u);
175           imuLevelChar1.writeValue(v);
176           imuLevelChar2.writeValue(w);
177           drawIMU(u,v,w);
178         }
179       }
180     }
```

*Should not advertise indefinitely

Demo: Android App

MainActivity.kt

```
147
148     override fun onCreate(savedInstanceState: Bundle?) {
149         super.onCreate(savedInstanceState)
150         setContentView(R.layout.activity_main)
151         if (BuildConfig.DEBUG) {
152             Timber.plant(Timber.DebugTree())
153         }
154         scan_button.setOnClickListener { if (isScanning) stopBleScan() else startBleScan() }
155         disconnect.setOnClickListener {  it: View!
156             if (device != null) {
157                 MyConnectionManager.teardownConnection(device!!)
158                 resetConnectionDisplay()
159             }
160         }
```

Button click listeners

- scan_button: start/stop scanning
- disconnect: end BLE connection to peripheral

MainActivity.kt

```
277     /*********************************************
278      * Callback bodies
279      *********************************************/
280
281     private val scanCallback = object : ScanCallback() {
282         override fun onScanResult(callbackType: Int, result: ScanResult) {
283             if (result.device.name == "SmartAlignment") {
284                 device_name.text = result.device.name
285                 device_address.text = result.device.address
286                 signal_strength.text = result.rssi.toString()+"dBm"
287
288                 device = result.device
289                 stopBleScan()
290                 MyConnectionManager.connect(result.device,  context: this@MainActivity)
291             }
```

Callback for BLE scan

- Connect to gatt server

Demo: Android App

Constants.kt

```kotlin
/ Copyright (C) 2021 The Android Open Source Project .../

package com.punchthrough.blestarterappandroid

import ...

/* Inertial Measurement Unit  */
const val ServiceUuid = "0000180f-0000-1000-8000-00805f9b34fb
const val CharUuid0 = "00002a19-0000-1000-8000-00805f9b34fb"
const val CharUuid1 = "00002a20-0000-1000-8000-00805f9b34fb"
const val CharUuid2 = "00002a21-0000-1000-8000-00805f9b34fb"

```

MyConnectionManager.kt

```kotlin

    fun readIMU(gatt: BluetoothGatt) {
        val imuServiceUuid = UUID.fromString(ServiceUuid)
        val imuCharUuid0 = UUID.fromString(CharUuid0)
        val imuCharUuid1 = UUID.fromString(CharUuid1)
        val imuCharUuid2 = UUID.fromString(CharUuid2)

        val imu0 = gatt.getService(imuServiceUuid)?.getCharacteristic(imuCharUuid0)
        val imu1 = gatt.getService(imuServiceUuid)?.getCharacteristic(imuCharUuid1)
        val imu2 = gatt.getService(imuServiceUuid)?.getCharacteristic(imuCharUuid2)

        if (imu0?.isReadable() == true) enableNotifications(gatt.device,imu0)
        if (imu1?.isReadable() == true) enableNotifications(gatt.device,imu1)
        if (imu2?.isReadable() == true) enableNotifications(gatt.device,imu2)
    }
```

Demo: Android App

Callback for characteristic change

- Update to sliding window and plot

MyConnectionManager.kt

```
460
461     override fun onCharacteristicChanged(
462         gatt: BluetoothGatt,
463         characteristic: BluetoothGattCharacteristic
464     ) {
465         with(characteristic) {  this: BluetoothGattCharacteristic
466             //Timber.i("Characteristic $uuid changed | value: ${value.toHexString()}")
467             //Timber.i("Characteristic $uuid changed | value: ${value.toNumString()}")
```

Conclusion

Thank you!

Subtitle