

「生理心理学的指標を用いた授業中の教師の認知負荷の把握」
(山森・長野・徳岡・草薙・大内, 2023) の
デバイス作成, 測定方法, データ処理について

目次

1	概要	1
2	デバイスの作成	4
2.1	皮膚コンダクタンスと末梢皮膚温の測定デバイス	4
2.2	身体の動きの大きさの測定デバイス	11
3	データ処理のスケッチ	13
3.1	皮膚コンダクタンスと末梢皮膚温の測定デバイス	13
3.2	身体の動きの大きさの測定デバイス	21
3.3	記録用パーソナルコンピュータ	33
4	測定方法	44
4.1	測定方法の概要	44
4.2	皮膚コンダクタンス測定のための調整	44
4.3	皮膚コンダクタンスと末梢皮膚温の測定	46
4.4	身体の動きの大きさの測定	47
4.5	測定データの記録とモニタリング	48
5	測定データの取り出しと可視化	50
5.1	測定データの取り出し	50
5.2	皮膚コンダクタンスと末梢皮膚温の変化の可視化	52

免責事項等

- 国立教育政策研究所や執筆者は、この文書の内容によって生じた損害等の一切の責任を負いかねます。
- この文書の内容は、正確性や安全性を保証するものではありません。

1 概要

この文書では、山森他 (2023) で用いた、皮膚コンダクタンス、末梢皮膚温、身体の動きの大きさを測定するデバイスと、測定データの処理について説明する。

測定デバイスは、皮膚コンダクタンスと末梢皮膚温を測定するもの (Figure 1)、身体の動きの大きさを測定するもの (Figure 2) の 2 種類である。これらのデバイスの回路、作成方法を、第 2 節で示す。マイクロコンピュータに書き込むスケッチは、第 3.1, 3.2 節で示す。これらのスケッチは、ArduinoIDE 1.8.19 を用いて作成した。

Figure 1

皮膚コンダクタンスと末梢皮膚温を測定するデバイス

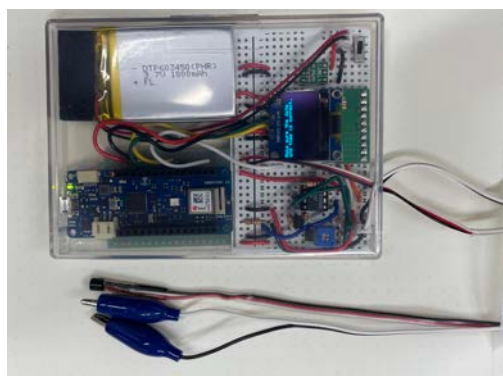


Figure 2

身体の動きの大きさを測定するデバイス



山森他 (2023) では、測定デバイスのマイクロコンピュータに M5StickC (M5Stack Tech-

nology) を用いているが、終売となったため、Arduino MKR WiFi 1010 (Arduino LLC) を用いた方法を示す。また、皮膚コンダクタンス測定では心電計用電極リード線を用いていたが、入手可能性を考慮し、ミノムシクリップを用いた方法を示す。

これら二つのデバイスで測定したデータは、Wi-Fi を経由して UDP で記録用パーソナルコンピュータに送信する。記録用パーソナルコンピュータには、Wi-Fi 通信が可能で、プログラミング言語であり開発環境である Processing が動作するものを用いる。

記録用パーソナルコンピュータは送信されたデータを処理し、csv 形式のファイルとして格納する。また、受信データをリアルタイムでプロットし、測定状況のモニタリングを行えるようにする (Figure 3)。これらの処理は、記録用パーソナルコンピュータにおいて、プログラミング言語であり開発環境である Processing で行う。Processing のスケッチは第 3.3 節で示す。用いた Processing のバージョンは 3.5.4 であった。

Figure 3

記録用パーソナルコンピュータでの測定状況のモニタリング



これらのデバイス及び記録用パーソナルコンピュータを用いた測定及びデータの記録の方法は、第 4 節で示す。デバイスの使用環境によっても異なるが、皮膚コンダクタンスと末梢皮膚温の測定デバイス、身体の動きの大きさの測定デバイスのリチウムイオン電池を満充電した場合、約 2 時間 30 分程度の連続測定が可能である。

記録用パーソナルコンピュータに格納された csv 形式のファイルに対する処理方法は任意である。第 5 節で、山森他 (2023) で用いた統計解析用プログラミング言語であり開発環境

である R のコードを示す。動作確認は、R version 4.1.2 (OS は macOS Monterey 12.6.2) で行った。

2 デバイスの作成

2.1 皮膚コンダクタンスと末梢皮膚温の測定デバイス

2.1.1 回路

皮膚コンダクタンスと末梢皮膚温を測定する回路では、リチウムイオン電池から供給される 3.7V の電圧を、DCDC コンバータ (AE-XCL102D503) で 5V に昇圧した。皮膚コンダクタンス測定のブリッジ回路、増幅回路は、Figure 4 のとおりであった。Figure 4 の b に示すように、R4、R5、VR 及び参加者の皮膚抵抗でブリッジ回路を構成し、その出力を計装アンプ LT1167 を用いて約 50 倍に増幅することで測定した。その際ブリッジ回路の電源電圧は美濃 (1986) を参考に、LM385Z-1.2 の出力を分圧し、およそ 0.5V とした。皮膚コンダクタンス測定の電源回路は Figure 4 の a に示すように、DCDC コンバータで昇圧した 5V の出力から TLE2426 により中間電圧 (2.5V) を取得し、それを仮想 GND とすることで ±2.5V の正負電源を構成し、計装アンプ LT1167 の駆動に用いた。

皮膚コンダクタンス測定部分、末梢皮膚温測定部分、ディスプレイと Arduino MKR WiFi 1010 の接続は、Figure 5 のとおりであった。

Figure 4

皮膚コンダクタンス測定の回路

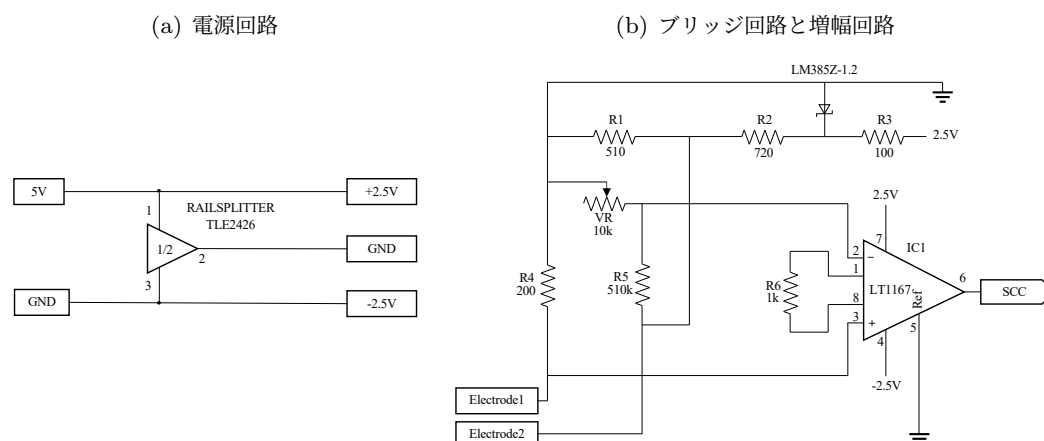
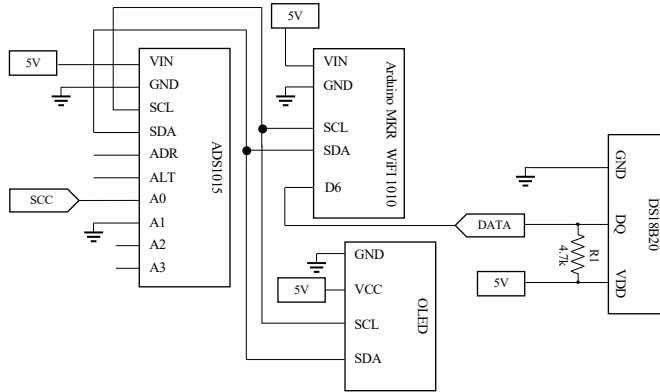


Figure 5

皮膚コンダクタンス測定部分，末梢皮膚温測定部分，ディスプレイと Arduino MKR WiFi 1010 の接続

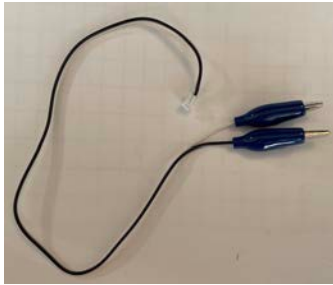


2.1.2 皮膚コンダクタンス測定の電極

Figure 6 のように，30cm の 2 心並列線の一方にミノムシクリップを，もう一方に PH コネクタハウジング (日本圧着端子製造 PHR-2) を取り付ける。極性はない。

Figure 6

皮膚コンダクタンス測定の電極とリード線



2.1.3 末梢皮膚温測定のセンサ

Figure 7 のように，40cm の 3 心並列線に，デジタル温度センサ (DS18B20) を取り付ける。ピンの短絡を防ぐために，あらかじめ黒，赤のリード線に収縮チューブを付け，GND に黒，DQQ に白，VDD に赤のリード線を半田付けする。さらに，センサ下部から 3 芯並列線全体に収縮チューブを取り付ける。

Figure 7

末梢皮膚温測定のためのセンサとリード線

(a) センサとリード線の接続



(b) リード線の処理



2.1.4 部品と配線

配線は、サンハヤトブレッドボード互換ユニバーサル基板 (UB-BRD01) 上で行った。基板上の部品配置は、Table 1, 2 のとおりである。

Table 1

部品と基板上的の配置（皮膚コンダクタンス測定部分）

種別	部品名	ピン	基板上的の位置		
回路電源	TLE2426CLP	1:IN	G30		
		2:COMMON	G29		
		3:OUT	G28		
		ジャンプワイヤ (赤)		+	L30
		ジャンプワイヤ (黒)		-	L29
	基準電圧 IC LM385Z-1.2	1:Anode	H28		
		2:Cathode	H27		
		3:NC	H26		
		抵抗 100 Ω (茶黒茶金)	L27 L+		
		抵抗 510 Ω (緑茶茶金)	I28 I26		
	抵抗 750 Ω (紫緑茶金)	G27 G26			
ブリッジ回路		ジャンプワイヤ (赤)	K26 B27		
		ジャンプワイヤ (黒)	K28 A29		
	抵抗 510k Ω (緑茶黄金)		D28 D27		
		抵抗 200 Ω (赤黒茶金)	D30 D29		
	半固定抵抗 10K Ω	左	F28		
		中央	E29		
		右	使用せず		
	PH コネクタベース付きポスト		A27		
			A28		
		ジャンプワイヤ (緑)	C28 H24		
	ジャンプワイヤ (緑)	C30 H23			
増幅回路	計装アンプ LT1167	1	G22		
		2	G23		
		3	G24		
		4	G25		
		5	F25		
		6	F24		
		7	F23		
		8	F22		
		ジャンプワイヤ (赤)	+ D23		
		ジャンプワイヤ (黒)	- L25		
AD 変換	AD コンバーター ADS1015	V	A9		
		G	A10		
		SCL	A11		
		SDA	A12		
		ADDR	A13		
		ALERT	A14		
		A0	A15		
		A1	A16		
		A2	A17		
		A3	A18		
	ジャンプワイヤ (赤)	+ E9			
	ジャンプワイヤ (黒)	- D10			

Table 2

部品と基板上的の配置（末梢皮膚温，電源，ディスプレイの部分）

種別	部品名	ピン	基板上的の位置
末梢皮膚温	デジタル温度センサ DS18B20 (センサを取り付けたリード線)	GND	-
		DQ	K19
電源	抵抗 4.7K Ω DCDC コンバータ AE-XCL102D503	VDD	+
			+ L19
		G	F7
		EN	F6
		I	F5
		I	G7
		G	G6
		O	G5
			- L6
			+ L5
			D3 D7
			E1 E5
			A2
			A3
	A4		
ディスプレイ	有機 EL ディスプレイ 0.96 インチ	PH コネクタベース付きポスト	C1
			C2
		GND	K15
		VCC	K14
電源	抵抗 4.7K Ω DCDC コンバータ AE-XCL102D503	SCL	K13
		SDA	K12
			I13 D11
			I12 E12
			+ L14
			- L15

ブレッドボード基板上的の配線の実際を示すと、Figure 8 のとおりである。注意した点を挙げると以下のとおりである。

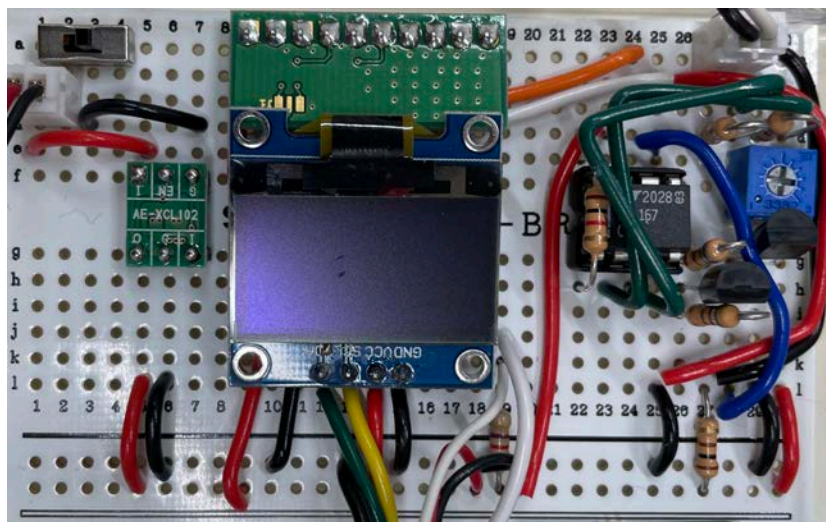
- ジャンプワイヤには、配線のしやすさと、ノイズを防ぐことを考慮し、導体径 0.65mm の単線導体使用の耐熱通信機器用ビニル電線 (協和ハーモネット H-PVC 0.65mm) を、適切な長さに切って用いた。
- 計装アンプ LT1167 は破損を防ぐために、IC ソケット (8 ピン) を用いて取り付けた。
- AD コンバータ ADS1015 には、ケースへの格納を考慮し、チップ実装面がブレッドボード基板に面するように細ピンヘッダを取り付け、基板に取り付けた。
- ディスプレイには、0.96 インチ 128×64 ドット有機 EL ディスプレイ (秋月電子通商) を用いた。ケースへの格納を考慮し、ADS1015 を取り付けた後に、ブレッドボード基板に取り付けた。
- ブレッドボード基板の C1, C2 に取り付けた PH コネクタベース付きポストは、リチ

ウムイオン電池の接続用である。C1 がリチウムイオン電池の + 極となるように、切り欠き部分をブレッドボード基板の外側を向くように取り付けた。

- ブレッドボード基板の A27, A28 に取り付けられた PH コネクタベース付きポストは、皮膚コンダクタンス測定の電極 (ミノムシクリップ) を取り付けたりード線を取り付けるためのものである。ケースへの格納を考慮し、切り欠き部分をブレッドボード基板の内側を向くように取り付けた。測定中のブレッドボード基板のランド剥離を防ぐために PH コネクタを用いて取り付けることとした。

Figure 8

ブレッドボード基板上の配線



マイクロコンピュータ (Arduino MKR WiFi 1010) と、部品等を実装したブレッドボード基板とは、Table 3 のとおりに接続した。電源はブレッドボード基板に接続したリチウムイオン電池から供給される。リチウムイオン電池は、過充電、過放電、過電流の保護回路内蔵の 3.7V, 1000mAh のものを、ブレッドボード基板の C1, C2 に取り付けられた PH コネクタベース付きポストに接続した。

Table 3

Arduino MKR WiFi 1010 とブレッドボード基板の接続

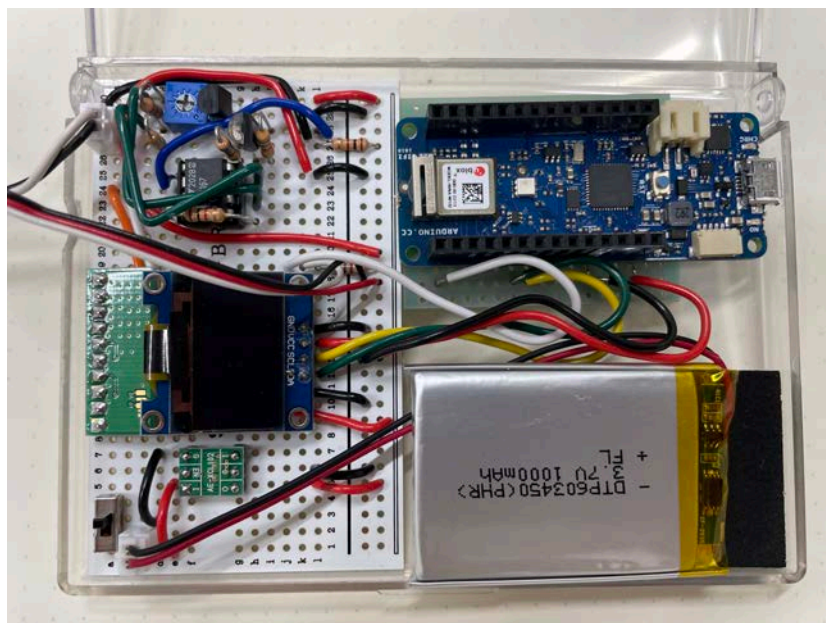
Arduino ピン	ブレッドボード基板
5V	+
GND	-
SCL	L13
SDA	L12
D6	J19

2.1.5 ケースへの格納

部品等を実装したブレッドボード基板、マイクロコンピュータ、リチウムイオン電池は、ABS樹脂ケース (西務良 112-TSS) に、Figure 9 のとおりに格納した。リチウムイオン電池を配置する部分には、ズレを防ぐために基板押さえスポンジ (タカチ電機工業 UR-55) を貼り付けた。皮膚コンダクタンス測定の電極を接続したリード線、末梢皮膚温測定のセンサを接続したリード線を引き出すために、ケース左上からの短辺 25mm の位置をヤスリで加工し、半円状の穴を開けた。

Figure 9

ABS樹脂ケースに格納



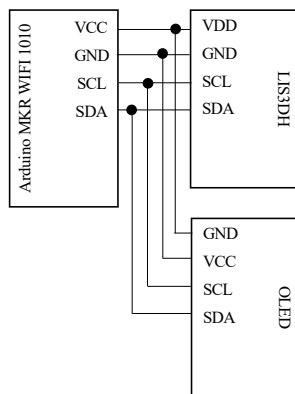
2.2 身体の動きの大きさの測定デバイス

2.2.1 回路

身体の動きの大きさの測定デバイスの回路は、Figure 10 のとおりである。マイクロコンピュータは、Arduino MKR WiFi 1010 を用いた。加速度センサは、3 軸加速度センサモジュール AE-LIS3DH(秋月電子通商) を、I2C で接続した。なお、入手不可の場合には同等品で代替可能だが、第節に示すスケッチの内容を変更する必要がある。ディスプレイには、0.96 インチ 128×64 ドット有機 EL ディスプレイ (秋月電子通商) を用いた。

Figure 10

身体の動きの大きさの測定デバイスの回路



2.2.2 デバイスの構造

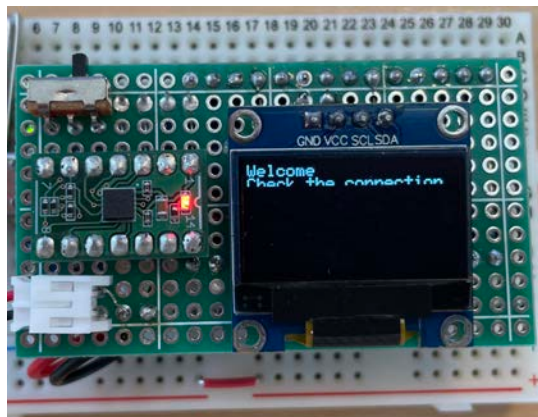
デバイス全体を小型化するために、Figure 11 に示す、Arduino MKR WiFi 1010 のシールドを作成した。加速度センサ、ディスプレイ、スライドスイッチ (1 回路 2 接点)、リチウムイオン電池接続用の PH コネクタベース付きポスト (サイド型) を、ユニバーサル基板に取り付けた。ユニバーサル基板には、Arduino MKR WiFi 1010 のシールドとして接続できるように、ピンヘッダを取り付けた。ユニバーサル基板は、56.5×32mm のもの (秋月電子通商 AE-FRSK-120-UV-TH) を用いた。電源には、過充電、過放電、過電流の保護回路内蔵の 3.7V、300mAh のリチウムイオン電池を用いた。リチウムイオン電池は、作成したシールドの PH コネクタベース付きポスト (サイド型) に接続した。作成したシールドの裏面には PH コネクタハウジング (日本圧着端子製造 PHR-2) 付きケーブルを取り付け、Arduino MKR WiFi 1010 のリチウムイオン電池用ピンに接続した。作成したシールド裏面の配線には、は

んだメッキ線と、外径 0.65mm の耐熱電線を用いた。

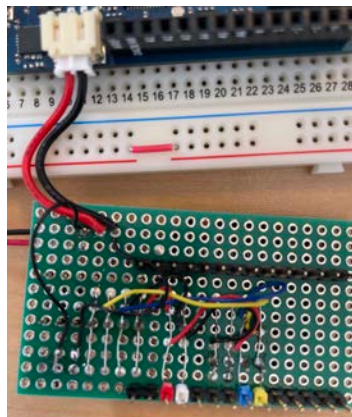
Figure 11

身体の動きの大きさの測定デバイスのシールド

(a) 表面



(b) 裏面



3 データ処理のスケッチ

3.1 皮膚コンダクタンスと末梢皮膚温の測定デバイス

3.1.1 スケッチ

皮膚コンダクタンスと末梢皮膚温を測定するデバイスのマイクロコンピュータ (Arduino MKR WiFi 1010) には、以下のスケッチを書き込む。

```
1  /* RTC 関係のライブラリ */
2  #include <SPI.h>
3  #include <WiFiNINA.h>
4  #include <WiFiUdp.h>
5  #include <RTCZero.h>
6
7  /* ディスプレイ関係のライブラリ */
8  #include <Wire.h>
9  #include <Adafruit_GFX.h>
10 #include <Adafruit_SSD1306.h>
11
12 /* SCTemp 関係のライブラリと変数設定 */
13 #include <Adafruit_ADS1X15.h>
14 #include <OneWire.h>
15 #include <DallasTemperature.h>
16
17 /* 皮膚コンダクタンスと末梢皮膚温測定のための変数とオブジェクトの宣言 */
18 #define ONE_WIRE_BUS 6 // Arduino に接続するポート番号
19 #define SENSER_BIT 9 // 9から12まで 9の方が取得間隔が短い
20
21 OneWire oneWire(ONE_WIRE_BUS);
22 DallasTemperature sensors(&oneWire);
23
24 float SC;
25 float tp;
26
27 // 抵抗値をマイクロジーメンズに変換するための値
28 const float slope = 0.028; // マイクロジーメンズ変換のための傾き
29 const float intercept = -2.026; // マイクロジーメンズ変換のための切片
30
```

```

31  const int sampling_period = 140; // サンプリングレート (7.14Hz)
32  int mil;
33
34  Adafruit_ADS1015 ads;
35  float multiplier;
36
37  /* RTC オブジェクトの作成 */
38  RTCZero rtc;
39
40  /* ディスプレイの設定 */
41  // 画面のサイズの設定
42  #define SCREEN_WIDTH (128)
43  #define SCREEN_HEIGHT (64)
44
45  // ディスプレイのアドレス (データシートから)
46  #define SCREEN_ADDRESS (0x3C)
47
48  // ディスプレイ変数の宣言
49  Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire);
50
51  /* Wi-Fi の設定 */
52  int status = WL_IDLE_STATUS;
53  char ssid[] = "XXXXXXXXXX";
54  char pass[] = "xxxxxxxxxx";
55
56  const char* saddr = "XXX.XXX.XXX.XXX";
57
58  const int sport = 55999;
59  const int kport = 5557;
60
61  char SCtpdat[100] = "";
62
63  IPAddress timeServer(129, 6, 15, 28);
64  const int NTP_PACKET_SIZE = 48;
65  byte packetBuffer[NTP_PACKET_SIZE];
66
67  WiFiUDP Udp;
68
69  /* RTC 関係の変数の宣言 */

```

```

70  int date_y;
71  int date_m;
72  int date_d;
73  int time_h;
74  int time_m;
75  int time_s;
76  int time_s_before;
77
78  /* void setup (一度だけ走らせるコード) */
79  void setup(){
80      /* 皮膚コンダクタンスと末梢皮膚温測定関係の設定 */
81      Serial.begin(115200);
82
83      sensors.setResolution(SENSER_BIT);
84
85      ads.setGain(GAIN_TWO); multiplier= 1.0F;
86      ads.begin();
87
88      /* ディスプレイの設定 */
89      if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
90          Serial.println(F("SSD1306 can not allocate memory!"));
91          return;
92      }
93      display.clearDisplay();
94      display.setTextSize(1);
95      display.setTextColor(WHITE);
96      display.setCursor(0, 0);
97      display.println("Welcome");
98      display.println("Check the connection.");
99      display.println("Check date and time");
100     display.println("settings.");
101
102     /* Wi-Fi 接続の確認*/
103     if (WiFi.status() == WL_NO_MODULE) {
104         display.println("Communication with WiFi module failed!");
105         display.display();
106         while (true);
107     }
108

```

```

109     String fv = WiFi.firmwareVersion();
110     if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
111         Serial.println("Please upgrade the firmware");
112     }
113
114     display.print("Connecting to SSID: ");
115     display.print(ssid);
116     while (status != WL_CONNECTED) {
117         display.print(".");
118         display.display();
119         status = WiFi.begin(ssid, pass);
120         delay(1000);
121     }
122
123     display.clearDisplay();
124     display.setCursor(0, 0);
125     display.println("Connected to WiFi");
126     display.display();
127     printWifiStatus();
128
129     /* NTP サーバーへの接続と時刻の設定 */
130     display.println("Connecting to NTP server...");
131     display.display();
132     Udp.begin(kport);
133     rtc.begin();
134
135     while(Udp.parsePacket()==0) {
136         sendNTPpacket(timeServer);
137         display.print(".");
138         display.display();
139         delay(2000);
140     }
141
142     display.clearDisplay();
143     display.setCursor(0, 0);
144     display.println("Packet received");
145     display.display();
146     Udp.read(packetBuffer, NTP_PACKET_SIZE);
147

```

```

148     unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
149     unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
150     unsigned long secsSince1900 = highWord << 16 | lowWord;
151
152     // Unix 時間は 1970 年 1 月 1 日(1900年 1月 1日から 2208988800秒後)
153     const unsigned long seventyYears = 2208988800UL;
154     // 70年引く
155     unsigned long epoch = secsSince1900 - seventyYears;
156     rtc.setEpoch(epoch + 32400); // 9時間分 (9x60x60)の時差を修正
157
158     print2digits(rtc.getYear());
159     display.print("/");
160     print2digits(rtc.getMonth());
161     display.print("/");
162     print2digits(rtc.getDay());
163     display.print("_");
164
165     print2digits(rtc.getHours());
166     display.print(":");
167     print2digits(rtc.getMinutes());
168     display.print(":");
169     print2digits(rtc.getSeconds());
170     display.println();
171     display.display();
172
173     /* 時間情報を変数に格納 */
174     date_y = rtc.getYear();
175     date_m = rtc.getMonth();
176     date_d = rtc.getDay();
177     time_h = rtc.getHours();
178     time_m = rtc.getMinutes();
179     time_s = rtc.getSeconds();
180     time_s_before = time_s;
181 }
182
183 /* void loop (くり返し走らせるコード) */
184 void loop() {
185     while(millis() < (mil + sampling_period)){ // サンプリングレートの時間が過ぎ
        まで待つ

```

```

186     }
187     mil = millis(); // 待ち時間終了直後の経過時間取得 サンプリングレート判定用なので
        processing に送信しない
188
189     int16_t results;
190     int16_t results_t;
191     results = ads.readADC_Differential_0_1();
192
193     sensors.requestTemperatures(); // 温度取得要求
194     tp = sensors.getTempCByIndex(0); // 温度の取得
195
196     int sctempmil = millis(); // データ取得後の経過時間取得
        processing に送信するデータはこちらの経過時間
197     SC = results * multiplier * slope + intercept;
198
199     Serial.print(sctempmil);
200     Serial.print(",");
201     Serial.print(SC);
202     Serial.print(",");
203     Serial.println(tp);
204
205     // 時間の確認
206     time_s = rtc.getSeconds();
207     if (time_s < time_s_before)
208     {
209         time_m = time_m + 1;
210         if (time_m >= 60)
211         {
212             time_m = time_m - 60;
213             time_h = time_h + 1;
214         }
215     }
216     time_s_before = time_s;
217
218     sprintf(SCTpdat, "SCtemp!,%02d/%02d/%02d_%02d:%02d:%02d,%10d,%.3f
        ,%4.1f,,,",date_y, date_m, date_d, time_h, time_m, time_s,
        sctempmil, SC, tp);
219     Udp.beginPacket(saddr, sport);
220     Udp.write(SCTpdat);

```

```

221     Udp.endPacket();
222     timedisplay();
223 }
224
225 /* NTP のパケットをタイムサーバーに送るための関数 */
226 unsigned long sendNTPpacket(IPAddress& address) {
227     // set all bytes in the buffer to 0
228     memset(packetBuffer, 0, NTP_PACKET_SIZE);
229     packetBuffer[0] = 0b11100011;
230     packetBuffer[1] = 0;
231     packetBuffer[2] = 6;
232     packetBuffer[3] = 0xEC;
233     packetBuffer[12] = 49;
234     packetBuffer[13] = 0x4E;
235     packetBuffer[14] = 49;
236     packetBuffer[15] = 52;
237
238     Udp.beginPacket(address, 123);
239     Udp.write(packetBuffer, NTP_PACKET_SIZE);
240     Udp.endPacket();
241 }
242
243 /* Wi-Fi の状況を有機 EL ディスプレイに表示する関数 */
244 void printWifiStatus() {
245     display.print("SSID:");
246     display.println(WiFi.SSID());
247
248     IPAddress ip = WiFi.localIP();
249     display.print("IP Address:");
250     display.println(ip);
251     display.display();
252 }
253
254 /* 時刻を 2桁で表示するための関数 */
255 void print2digits(int number) {
256     if (number < 10) {
257         display.print("0");
258     }
259     display.print(number);

```

```

260 }
261
262 /* ディスプレイに必要な情報を表示する関数 */
263 void timedisplay(){
264     display.clearDisplay();
265     display.setTextSize(1);
266     display.setCursor(0, 0);
267
268     print2digits(date_y);
269     display.print("/");
270     print2digits(date_m);
271     display.print("/");
272     print2digits(date_d);
273     display.print("_");
274
275     print2digits(time_h);
276     display.print(":");
277     print2digits(time_m);
278     display.print(":");
279     print2digits(time_s);
280     display.println();
281
282     display.setTextSize(1);
283     display.print("SC:");
284     display.println(SC);
285
286     display.print("Temp:");
287     display.println(tp);
288
289     display.display();
290 }

```

3.1.2 解説

1-15行 必要なライブラリの読み込み

17-35行 皮膚コンダクタンスと末梢皮膚温測定のための変数宣言 (27-29行目は抵抗値をマイクロジーメンズに変換するための値)

37-38行 時刻制御のオブジェクトを定義

40-49 行 有機 EL ディスプレイのアドレスなどの設定

51-59 行 Wi-Fi の設定。53, 54 行目に Wi-Fi ルーターの SSID とパスワード, 56 行目に記録用パーソナルコンピュータの IP アドレス, 58-59 行目のポート番号は任意だが, 測定データの記録とモニタリングの処理を行う Processing と一致させる

61 行 送信するデータを格納する変数の宣言

63-67 行 時刻を取得する NTP サーバと UDP の設定

69-76 行 時刻データ格納用変数の宣言

78-181 行 温度センサの設定は DallasTemperature, 皮膚コンダクタンス測定関係の設定は Adafruit_ADS1X15, ディスプレイの設定は Adafruit_SSD1306, Wi-Fi 接続の確認と NTP サーバへの接続と時刻の設定は WiFiNINA 及び RTCZero のライブラリのサンプルスケッチを参照して記述

183-223 行 皮膚コンダクタンスと末梢皮膚温データの取得, 皮膚コンダクタンスの単位をマイクロジーメンスに変換, リアルタイムクロックから得られる時間データの処理, UDP を用いた時刻, 経過時間, 皮膚コンダクタンスと末梢皮膚温の送信, Arduino IDE のシリアルモニタで確認できるようにシリアルポートにデータを送信, 関数を用いて有機 EL ディスプレイに時刻等を表示

225-241 行 NTP のパケットをタイムサーバに送るための関数を, WiFiNINA ライブラリのサンプルスケッチを参照して記述

243-252 行 Wi-Fi の状況を有機 EL ディスプレイに表示するための関数を, WiFiNINA ライブラリのサンプルスケッチを参照して記述

254-260 行 時刻を 2 桁で表示するための関数を, RTCZero ライブラリのサンプルスケッチを参照して記述

262-290 行 有機 EL ディスプレイに, 時刻, 皮膚コンダクタンス, 末梢皮膚温を表示するための関数

3.2 身体の動きの大きさの測定デバイス

3.2.1 スケッチ

身体の動きの大きさを測定するデバイスのマイクロコンピュータ (Arduino MKR WiFi 1010) には, 以下のスケッチを書き込む。

```
1  /* RTC 関係のライブラリ */
2  #include <SPI.h>
3  #include <WiFiNINA.h>
4  #include <WiFiUdp.h>
```

```

5  #include <RTCZero.h>
6
7  /* ディスプレイ関係のライブラリ */
8  #include <Wire.h>
9  #include <Adafruit_GFX.h>
10 #include <Adafruit_SSD1306.h>
11
12 /* 加速度センサ関係のライブラリ */
13 #include <Adafruit_LIS3DH.h>
14
15 /* RTC オブジェクトの作成 */
16 RTCZero rtc;
17
18 /* 加速度センサのI2C */
19 #define LIS3DH_ADDRESS 0x18
20 Adafruit_LIS3DH accel = Adafruit_LIS3DH();
21
22 /* ディスプレイの設定 */
23 // 画面のサイズ
24 #define SCREEN_WIDTH (128)
25 #define SCREEN_HEIGHT (64)
26
27 //ディスプレイのアドレス (データシートから)
28 #define SCREEN_ADDRESS (0x3C)
29
30 // ディスプレイ変数の宣言
31 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire);
32
33 /* Wi-Fi の設定 */
34 int status = WL_IDLE_STATUS;
35 char ssid[] = "XXXXXXXXXX";
36 char pass[] = "xxxxxxxxxx";
37
38 const char* saddr = "XXX.XXX.XXX.XXX";
39
40 const int sport = 55998;
41 const int kport = 5556;
42
43 char Hzdat[100] = "";

```

```

44
45  IPAddress timeServer(129, 6, 15, 28);
46  const int NTP_PACKET_SIZE = 48;
47  byte packetBuffer[NTP_PACKET_SIZE];
48
49  WiFiUDP Udp;
50
51  /* FFTに必要なライブラリと変数の宣言 */
52  #include <arduinoFFT.h>
53
54  #define SAMPLING_FREQUENCY 50
55  const uint16_t FFTsamples = 256; // サンプル数は2のべき乗
56  double vReal[FFTsamples]; // vReal[]にサンプリングしたデータを入れる
57  double vImag[FFTsamples];
58  double vLog[FFTsamples];
59  arduinoFFT FFT = arduinoFFT(vReal, vImag, FFTsamples,
        SAMPLING_FREQUENCY); // FFT オブジェクトを作る
60  unsigned int sampling_period_us;
61
62  //unsigned long microseconds;
63  float indextoHz; // 周波数スペクトルのインデックス番号をHzに修正するための係数
64  int fftloop; // 周波数スペクトルの最大値検索を5Hzまでに留めるための変数
65  int peak; // 周波数スペクトルの最大値のインデックス番号を格納するための変数
66  int mil; // サンプリング周波数を維持するための経過時間を格納するための変数
67  int milf; // FFT実施後の経過時間を格納するための変数
68  int mila; // 合成加速度データ取得後の経過時間を格納するための変数
69
70  /* 加速度センサの変数の宣言 */
71  float accX = 0.0F;
72  float accY = 0.0F;
73  float accZ = 0.0F;
74  float acc;
75
76  /* RTC 関係の変数の宣言 */
77  int date_y;
78  int date_m;
79  int date_d;
80  int time_h;
81  int time_m;

```

```

82  int time_s;
83  int time_s_before;
84
85  /* データ取得のための関数 */
86  void sample(int nsamples) {
87      int i;
88      for (int i = 0; i < nsamples; i++) {
89          while(millis() < (mil + sampling_period_us)){
90              }
91
92          mil = millis();
93          mila = millis();
94          accel.read();
95          accX = accel.x_g;
96          accY = accel.y_g;
97          accZ = accel.z_g;
98
99          acc = sqrt(sq(accX)+sq(accY)+sq(accZ));
100
101          // 時間の確認
102          time_s = rtc.getSeconds();
103          if (time_s < time_s_before)
104              {
105                  time_m = time_m + 1;
106                  if (time_m >= 60)
107                      {
108                          time_m = time_m - 60;
109                          time_h = time_h + 1;
110                      }
111              }
112          time_s_before = time_s;
113          sprintf(Hzdat, "acc!,%02d/%02d/%02d_%02d:%02d:%02d,%10d,%5.2f,,,,,0,",
114              date_y, date_m, date_d, time_h, time_m, time_s, mila, acc);
115
116          Udp.begin(kport);
117          Udp.beginPacket(saddr, sport);
118          Udp.write(Hzdat);
119          Udp.endPacket();
120          Serial.print(mila);

```

```

121     Serial.print(",");
122     Serial.print(acc);
123     Serial.print(",");
124     Serial.print(peak);
125     Serial.print(",");
126     float hz;
127     hz = (float)peak * indextoHz;
128     Serial.println(float(hz));
129
130     vReal[i] = acc;
131     vLog[i] = acc;
132     vImag[i] = 0;
133
134     }
135 }
136
137 /* void setup (一度だけ走らせるコード) */
138 void setup(){
139     /* 加速度センサの設定 */
140     Serial.begin(115200);
141     accel.begin(LIS3DH_ADDRESS);
142     accel.setClick(0, 0);
143     accel.setRange(LIS3DH_RANGE_8_G);
144     accel.setDataRate(LIS3DH_DATARATE_400_HZ);
145
146     /* ディスプレイの設定 */
147     if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
148         Serial.println(F("SSD1306 can not allocate memory!"));
149         return;
150     }
151     display.clearDisplay();
152     display.setTextSize(1);
153     display.setTextColor(WHITE);
154     display.setCursor(0, 0);
155     display.println("Welcome");
156     display.println("Check the connection.");
157     display.println("Check date and time");
158     display.println("settings.");
159

```

```

160  /* Wi-Fi 接続の確認*/
161  if (WiFi.status() == WL_NO_MODULE) {
162      display.println("Communication_with_WiFi_module_failed!");
163      display.display();
164      while (true);
165  }
166
167  String fv = WiFi.firmwareVersion();
168  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
169      Serial.println("Please_upgrade_the_firmware");
170  }
171
172  display.print("Connecting_to_SSID:");
173  display.print(ssid);
174  while (status != WL_CONNECTED) {
175      display.print(".");
176      display.display();
177      status = WiFi.begin(ssid, pass);
178      delay(1000);
179  }
180
181  display.clearDisplay();
182  display.setCursor(0, 0);
183  display.println("Connected_to_WiFi");
184  display.display();
185  printWifiStatus();
186
187  /* NTP サーバーへの接続と時刻の設定 */
188  display.println("Connecting_to_NTP_server...");
189  display.display();
190  Udp.begin(kport);
191  rtc.begin();
192
193  while(Udp.parsePacket()==0) {
194      sendNTPpacket(timeServer); // send an NTP packet to a time server
195      display.print(".");
196      display.display();
197      delay(2000);
198  }

```

```

199
200     display.clearDisplay();
201     display.setCursor(0, 0);
202     display.println("Packet received");
203     display.display();
204     Udp.read(packetBuffer, NTP_PACKET_SIZE);
205
206     unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
207     unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
208     unsigned long secsSince1900 = highWord << 16 | lowWord;
209
210     // Unix 時間は 1970 年 1 月 1 日(1900年 1月 1日から 2208988800秒後)
211     const unsigned long seventyYears = 2208988800UL;
212     // 70年引く
213     unsigned long epoch = secsSince1900 - seventyYears;
214     rtc.setEpoch(epoch + 32400); // 9時間分 (9x60x60)の時差を修正
215     display.println("Complete.");
216     display.clearDisplay();
217
218     print2digits(rtc.getYear());
219     display.print("/");
220     print2digits(rtc.getMonth());
221     display.print("/");
222     print2digits(rtc.getDay());
223     display.print("_");
224
225     print2digits(rtc.getHours());
226     display.print(":");
227     print2digits(rtc.getMinutes());
228     display.print(":");
229     print2digits(rtc.getSeconds());
230     display.println();
231     display.display();
232
233     /* 時間情報を変数に格納 */
234     date_y = rtc.getYear();
235     date_m = rtc.getMonth();
236     date_d = rtc.getDay();
237     time_h = rtc.getHours();

```

```

238     time_m = rtc.getMinutes();
239     time_s = rtc.getSeconds();
240     time_s_before = time_s;
241
242     /* FFT 関連の変数の設定 */
243     sampling_period_us = round(1000*(1.0/SAMPLING_FREQUENCY));
244     indextoHz = (float)SAMPLING_FREQUENCY / FFTsamples;
245     fftloop = 5 / indextoHz + 1; //5
           Hz までの周波数スペクトル最大インデックス番号を計算
246     //FFT
247     mil = millis();
248 }
249
250 /* void loop (くり返し走らせるコード) */
251 void loop() {
252     sample(FFTsamples);
253     DCRemoval(vReal, FFTsamples);
254     FFT.Windowing(FFT_WIN_TYP_HAMMING, FFT_FORWARD); // 窓関数
255     FFT.Compute(FFT_FORWARD); // FFT 処理(複素数で計算)
256     FFT.ComplexToMagnitude(); // 複素数を実数に変換
257
258     int i;
259     float peakm = 0;
260     for(i = 0; i < fftloop; i++){ // 5Hz 相当までしか判定しない
261         if(peakm<=vReal[i]){
262             peakm=vReal[i];
263             peak = i;
264         }
265     }
266
267     milf = millis();
268     Serial.print("Data:");
269     Serial.print(milf);
270     Serial.print(",");
271     Serial.println(peak * indextoHz);
272     // 時間の確認
273     time_s = rtc.getSeconds();
274     if (time_s < time_s_before)
275     {

```

```

276         time_m = time_m + 1;
277         if (time_m >= 60)
278         {
279             time_m = time_m - 60;
280             time_h = time_h + 1;
281         }
282     }
283     time_s_before = time_s;
284
285     sprintf(Hzdat, "aFFT!,%02d/%02d/%02d,%02d:%02d:%02d,%10d,,,.2f,%3d
        ,%3d",date_y, date_m, date_d, time_h, time_m, time_s, milf, peak
        * indextoHz, SAMPLING_FREQUENCY, FFTsamples);
286     Udp.beginPacket(saddr, sport);
287     Udp.write(Hzdat);
288     Udp.endPacket();
289     timedisplay();
290 }
291
292 /* NTP のパケットをタイムサーバーに送るための関数 */
293 unsigned long sendNTPpacket(IPAddress& address) {
294     memset(packetBuffer, 0, NTP_PACKET_SIZE);
295     packetBuffer[0] = 0b11100011;
296     packetBuffer[1] = 0;
297     packetBuffer[2] = 6;
298     packetBuffer[3] = 0xEC;
299     packetBuffer[12] = 49;
300     packetBuffer[13] = 0x4E;
301     packetBuffer[14] = 49;
302     packetBuffer[15] = 52;
303
304     Udp.beginPacket(address, 123);
305     Udp.write(packetBuffer, NTP_PACKET_SIZE);
306     Udp.endPacket();
307 }
308
309 /* 直流成分の除去を行う関数 */
310 void DCRemoval(double *vData, uint16_t samples) {
311     double mean = 0;
312     for (uint16_t i = 1; i < samples; i++) {

```

```

313         mean += vData[i];
314     }
315     mean /= samples;
316     for (uint16_t i = 1; i < samples; i++) {
317         vData[i] -= mean;
318     }
319 }
320
321 /* Wi-Fi の状況を有機 EL ディスプレイに表示する関数 */
322 void printWifiStatus() {
323     display.print("SSID:");
324     display.println(WiFi.SSID());
325
326     IPAddress ip = WiFi.localIP();
327     display.print("IP Address:");
328     display.println(ip);
329     display.display();
330 }
331
332 /* 時刻を 2桁で表示するための関数 */
333 void print2digits(int number) {
334     if (number < 10) {
335         display.print("0");
336     }
337     display.print(number);
338 }
339
340 /* ディスプレイに必要な情報を表示する関数 */
341 void timedisplay(){
342     display.clearDisplay();
343     display.setTextSize(1);
344     display.setCursor(0, 0);
345
346     print2digits(date_y);
347     display.print("/");
348     print2digits(date_m);
349     display.print("/");
350     print2digits(date_d);
351     display.print("_");

```

```

352
353     print2digits(time_h);
354     display.print(":");
355     print2digits(time_m);
356     display.print(":");
357     print2digits(time_s);
358     display.println();
359
360     display.setTextSize(1);
361     display.print("ACC_X:");
362     display.print(accX);
363     display.println("G");
364
365     display.print("ACC_Y:");
366     display.print(accY);
367     display.println("G");
368
369     display.print("ACC_Z:");
370     display.print(accZ);
371     display.println("G");
372
373     display.print("S_ACC:");
374     display.print(acc);
375     display.println("G");
376     display.print("FFT:");
377     display.print(peak * indextoHz);
378     display.print("HZ");
379     display.println();
380
381     display.display();
382 }

```

3.2.2 解説

1-13行 必要なライブラリの読み込み

15-16行 時刻制御のオブジェクトを定義

18-20行 I2C 接続の加速度センサのアドレスとオブジェクトを定義

22-31行 有機 EL ディスプレイのアドレスなどの設定

33-41 行 Wi-Fi の設定。35, 36 行目に Wi-Fi ルーターの SSID とパスワード, 38 行目に記録用パーソナルコンピュータの IP アドレス, 40-41 行目のポート番号は任意だが, 測定データの記録とモニタリングの処理を行う Processing と一致させる

43 行 送信するデータを格納する変数の宣言

45-49 行 時刻を取得する NTP サーバと UDP の設定

51-74 行 加速度及び周波数解析結果の取得のためのライブラリの読み込みと変数の宣言

76-83 行 時刻データ格納用変数の宣言

85-135 行 3 軸加速度データの取得, 合成加速度の計算, リアルタイムクロックから得られる時間データの処理, UDP を用いた時刻, 経過時間, 合成加速度の送信, Arduino IDE のシリアルモニタで確認できるようシリアルポートにもデータを送信, 周波数解析を行うための変数にデータを格納, これらを定められた回数 (256 回) 繰り返す関数を定義

137-248 行 加速度センサの設定は Adafruit_LIS3DH, ディスプレイの設定は Adafruit_SSD1306, Wi-Fi 接続の確認と NTP サーバへの接続と時刻の設定は WiFiNINA 及び RTCZero, 周波数解析 (高速フーリエ変換 [FFT]) 関係は ArduinoFFT のライブラリのサンプルスケッチを参照して記述

250-290 行 加速度データ取得の関数を実行, 直流成分を除去した上での FFT 処理, 定められた範囲 (5Hz まで) での周波数スペクトルのピークを判定, Arduino IDE のシリアルモニタで確認できるようシリアルポートにデータ送信, リアルタイムクロックから得られる時間データ処理, UDP を用いた時刻, 経過時間, ピーク周波数データの送信, 関数を用いて有機 EL ディスプレイに時刻等を表示

292-307 行 NTP のパケットをタイムサーバに送るための関数を, WiFiNINA ライブラリのサンプルスケッチを参照して記述

309-319 行 FFT を行うための合成加速度のサンプルに対して直流成分を除去するための関数

321-330 行 Wi-Fi の状況を有機 EL ディスプレイに表示するための関数を, WiFiNINA ライブラリのサンプルスケッチを参照して記述

332-338 行 時刻を 2 桁で表示するための関数を, RTCZero ライブラリのサンプルスケッチを参照して記述

340-382 行 有機 EL ディスプレイに, 時刻, 3 軸の加速度, 合成加速度, ピーク周波数を表示するための関数

3.3 記録用パーソナルコンピュータ

3.3.1 スケッチ

記録用パーソナルコンピュータで Processing を起動し、以下のスケッチを実行する。

```
1  /* UDP ライブラリのインポート */
2  import hypermedia.net.*;
3
4  /* データ受信の設定 */
5  // 身体の動きの大きさの測定デバイスのArduino スケッチの sport と一致させる
6  int port1 = 55998;
7  // 皮膚コンダクタンスなどの測定デバイスのArduino スケッチの sport と一致させる
8  int port2 = 55999;
9
10 /* 変数の指定*/
11 int mil;
12 int sctempmil;
13 int accmilst;
14 int sctempmilst;
15 float acc;
16 float fft;
17 float sc;
18 float tp;
19 String filename1;
20
21 /* オブジェクトの宣言 */
22 graphMonitor_acc accGraph;
23 graphMonitor_fft fftGraph;
24 graphMonitor_temp tempGraph;
25 graphMonitor_sc scGraph;
26 UDP udp1;
27 UDP udp2;
28 PrintWriter output1;
29
30 /* void setup (一度だけ走らせるコード) */
31 void setup() {
32     /* リアルタイムグラフ関係*/
33     size(1366,768);
```

```

34   PFont font = createFont("Arial",48,true);
35   textFont(font);
36   frameRate(60);
37   smooth();
38   accGraph = new graphMonitor_acc("Acc", 794, 50, 500, 300);
39   fftGraph = new graphMonitor_fft("Hz", 122, 50, 500, 300);
40   scGraph = new graphMonitor_sc("SC", 794, 418, 500, 300);
41   tempGraph = new graphMonitor_temp("Temp", 122, 418, 500, 300);
42
43   /* データの格納関係*/
44   filename1 = nf(year(),2)+nf(month(),2)+nf(day(),2)+nf(hour(),2)+nf(
         minute(),2)+nf(second(),2)+"_log1.csv";
45   output1 = createWriter(filename1);
46
47   udp1 = new UDP( this, 55998 );
48   udp1.listen( true );
49   udp2 = new UDP( this, 55999 );
50   udp2.listen( true );
51
52   accmilst = 0;
53   sctempmilst = 0;
54 }
55
56 /* void draw (くり返し走らせるコード) */
57 void draw() {
58   /* リアルタイムグラフの描画と経過時間の表示*/
59   background(245,245,245);
60   accGraph.graphDraw_acc(acc);
61   fftGraph.graphDraw_fft(fft);
62   tempGraph.graphDraw_temp(tp);
63   scGraph.graphDraw_sc(sc);
64   text(int(mil),100,18);
65 }
66
67 /* キーの操作関係 e を押して終了させる*/
68 void keyPressed() {
69   if( key == 'e' ){
70     output1.flush();
71     output1.close();

```

```

72     delay(1000);
73     exit();
74 }
75 }
76
77 /* 合成加速度のリアルタイムグラフ表示のクラス*/
78 class graphMonitor_acc {
79     String TITLE;
80     int X_POSITION, Y_POSITION;
81     int X_LENGTH, Y_LENGTH;
82     float [] y1;
83     float maxRange;
84     float minRange;
85     graphMonitor_acc(String _TITLE, int _X_POSITION, int _Y_POSITION,
86         int _X_LENGTH, int _Y_LENGTH) {
87         TITLE = _TITLE;
88         X_POSITION = _X_POSITION;
89         Y_POSITION = _Y_POSITION;
90         X_LENGTH = _X_LENGTH;
91         Y_LENGTH = _Y_LENGTH;
92         y1 = new float[X_LENGTH];
93         for (int i = 0; i < X_LENGTH; i++) {
94             y1[i] = 0;
95         }
96     }
97
98 /* 合成加速度のリアルタイムグラフ表示の関数*/
99     void graphDraw_acc(float _y1) {
100         y1[X_LENGTH - 1] = _y1;
101         for (int i = 0; i < X_LENGTH - 1; i++) {
102             y1[i] = y1[i + 1];
103         }
104         maxRange = 2;
105         for (int i = 0; i < X_LENGTH - 1; i++) {
106             maxRange = (abs(y1[i]) > maxRange ? abs(y1[i]) : maxRange);
107         }
108         minRange = 0.5;
109         for (int i = 0; i < X_LENGTH - 1; i++) {
110             minRange = (abs(y1[i]) < minRange ? abs(y1[i]) : minRange);

```

```

110     }
111
112     pushMatrix();
113
114     translate(X_POSITION, Y_POSITION);
115     fill(255);
116     stroke(130);
117     strokeWeight(1);
118     rect(0, 0, X_LENGTH, Y_LENGTH);
119     line(0, Y_LENGTH / 2, X_LENGTH, Y_LENGTH / 2);
120     textSize(25);
121     fill(51,51,204);
122     textAlign(LEFT, BOTTOM);
123     text(TITLE, 20, -5);
124     textSize(22);
125     textAlign(RIGHT);
126     text(nf(maxRange, 0, 1), -5, 18);
127     text(nf(minRange, 0, 1), -5, Y_LENGTH);
128
129     translate(0, Y_LENGTH / 2);
130     scale(1, -1);
131     strokeWeight(3);
132     for (int i = 0; i < X_LENGTH - 1; i++) {
133         stroke(0, 102, 204);
134         line(i, ((y1[i] - minRange) * Y_LENGTH / (maxRange - minRange)
135             )) - (Y_LENGTH / 2), i + 1, ((y1[i + 1] - minRange) *
136             Y_LENGTH / (maxRange - minRange)) - (Y_LENGTH / 2));
137     }
138     popMatrix();
139 }
140
141 /* 周波数のリアルタイムグラフ表示のクラス */
142 class graphMonitor_fft {
143     String TITLE;
144     int X_POSITION, Y_POSITION;
145     int X_LENGTH, Y_LENGTH;
146     float [] y1;
147     float maxRange;

```

```

147     graphMonitor_fft(String _TITLE, int _X_POSITION, int _Y_POSITION,
148         int _X_LENGTH, int _Y_LENGTH) {
149         TITLE = _TITLE;
150         X_POSITION = _X_POSITION;
151         Y_POSITION = _Y_POSITION;
152         X_LENGTH = _X_LENGTH;
153         Y_LENGTH = _Y_LENGTH;
154         y1 = new float[X_LENGTH];
155         for (int i = 0; i < X_LENGTH; i++) {
156             y1[i] = 0;
157         }
158     }
159     /* 周波数のリアルタイムグラフ表示の関数 */
160     void graphDraw_fft(float _y1) {
161         y1[X_LENGTH - 1] = _y1;
162         for (int i = 0; i < X_LENGTH - 1; i++) {
163             y1[i] = y1[i + 1];
164         }
165         maxRange = 5;
166         for (int i = 0; i < X_LENGTH - 1; i++) {
167             maxRange = (abs(y1[i]) > maxRange ? abs(y1[i]) : maxRange);
168         }
169
170         pushMatrix();
171
172         translate(X_POSITION, Y_POSITION);
173         fill(255);
174         stroke(130);
175         strokeWeight(1);
176         rect(0, 0, X_LENGTH, Y_LENGTH);
177         line(0, Y_LENGTH / 2, X_LENGTH, Y_LENGTH / 2);
178         textSize(25);
179         fill(51,51,204);
180         textAlign(LEFT, BOTTOM);
181         text(TITLE, 20, -5);
182         textSize(22);
183         textAlign(RIGHT);
184         text(nf(maxRange, 0, 1), -5, 18);

```

```

185     text(0, -5, Y_LENGTH);
186
187     translate(0, Y_LENGTH / 2);
188     scale(1, -1);
189     strokeWidth(3);
190     for (int i = 0; i < X_LENGTH - 1; i++) {
191         stroke(0, 102, 204);
192         line(i, (y1[i] * Y_LENGTH / maxRange) - (Y_LENGTH / 2), i +
193             1, (y1[i + 1] * Y_LENGTH / maxRange) - (Y_LENGTH / 2));
194     }
195     popMatrix();
196 }
197
198 /* 抹消皮膚温のリアルタイムグラフ表示のクラス */
199 class graphMonitor_temp {
200     String TITLE;
201     int X_POSITION, Y_POSITION;
202     int X_LENGTH, Y_LENGTH;
203     float [] y1;
204     float maxRange;
205     float minRange;
206     graphMonitor_temp(String _TITLE, int _X_POSITION, int _Y_POSITION,
207         int _X_LENGTH, int _Y_LENGTH) {
208         TITLE = _TITLE;
209         X_POSITION = _X_POSITION;
210         Y_POSITION = _Y_POSITION;
211         X_LENGTH = _X_LENGTH;
212         Y_LENGTH = _Y_LENGTH;
213         y1 = new float[X_LENGTH];
214         for (int i = 0; i < X_LENGTH; i++) {
215             y1[i] = 0;
216         }
217     }
218
219 /* 抹消皮膚温のリアルタイムグラフ表示の関数 */
220 void graphDraw_temp(float _y1) {
221     y1[X_LENGTH - 1] = _y1;
222     for (int i = 0; i < X_LENGTH - 1; i++) {

```

```

222     y1[i] = y1[i + 1];
223 }
224 maxRange = 37;
225 for (int i = 0; i < X_LENGTH - 1; i++) {
226     maxRange = (abs(y1[i]) > maxRange ? abs(y1[i]) : maxRange);
227 }
228 minRange = 30;
229 for (int i = 0; i < X_LENGTH - 1; i++) {
230     minRange = (abs(y1[i]) < minRange ? abs(y1[i]) : minRange);
231 }
232
233     pushMatrix();
234
235     translate(X_POSITION, Y_POSITION);
236     fill(255);
237     stroke(130);
238     strokeWeight(1);
239     rect(0, 0, X_LENGTH, Y_LENGTH);
240     line(0, Y_LENGTH / 2, X_LENGTH, Y_LENGTH / 2);
241
242     textSize(25);
243     fill(51,51,204);
244     textAlign(LEFT, BOTTOM);
245     text(TITLE, 20, -5);
246     textSize(22);
247     textAlign(RIGHT);
248     text(nf(maxRange, 0, 1), -5, 18);
249     text(nf(minRange, 0, 1), -5, Y_LENGTH);
250     translate(0, Y_LENGTH / 2);
251     scale(1, -1);
252     strokeWeight(3);
253     for (int i = 0; i < X_LENGTH - 1; i++) {
254         stroke(0, 102, 204);
255         line(i, ((y1[i] - minRange) * Y_LENGTH / (maxRange - minRange
                )) - (Y_LENGTH / 2), i + 1, ((y1[i + 1] - minRange) *
                Y_LENGTH / (maxRange - minRange)) - (Y_LENGTH / 2));
256
257     }
258     popMatrix();

```

```

259     }
260 }
261
262 /* 皮膚コンダクタンスのリアルタイムグラフ表示のクラス */
263 class graphMonitor_sc {
264     String TITLE;
265     int X_POSITION, Y_POSITION;
266     int X_LENGTH, Y_LENGTH;
267     float [] y1;
268     float maxRange;
269     graphMonitor_sc(String _TITLE, int _X_POSITION, int _Y_POSITION,
270                     int _X_LENGTH, int _Y_LENGTH) {
271         TITLE = _TITLE;
272         X_POSITION = _X_POSITION;
273         Y_POSITION = _Y_POSITION;
274         X_LENGTH = _X_LENGTH;
275         Y_LENGTH = _Y_LENGTH;
276         y1 = new float[X_LENGTH];
277         for (int i = 0; i < X_LENGTH; i++) {
278             y1[i] = 0;
279         }
280     }
281
282 /* 皮膚コンダクタンスのリアルタイムグラフ表示の関数 */
283 void graphDraw_sc(float _y1) {
284     y1[X_LENGTH - 1] = _y1;
285     for (int i = 0; i < X_LENGTH - 1; i++) {
286         y1[i] = y1[i + 1];
287     }
288     maxRange = 30;
289     for (int i = 0; i < X_LENGTH - 1; i++) {
290         maxRange = (abs(y1[i]) > maxRange ? abs(y1[i]) : maxRange);
291     }
292
293     pushMatrix();
294
295     translate(X_POSITION, Y_POSITION);
296     fill(255);
297     stroke(130);

```

```

297     strokeWeight(1);
298     rect(0, 0, X_LENGTH, Y_LENGTH);
299     line(0, Y_LENGTH / 2, X_LENGTH, Y_LENGTH / 2);
300
301     textSize(25);
302     fill(51,51,204);
303     textAlign(LEFT, BOTTOM);
304     text(TITLE, 20, -5);
305     textSize(22);
306     textAlign(RIGHT);
307     text(nf(maxRange, 0, 0), -5, 18);
308     text(0, -5, Y_LENGTH);
309
310     translate(0, Y_LENGTH / 2);
311     scale(1, -1);
312     strokeWeight(3);
313     for (int i = 0; i < X_LENGTH - 1; i++) {
314         stroke(0, 102, 204);
315         line(i, (y1[i] * Y_LENGTH / maxRange) - (Y_LENGTH / 2), i +
316             1, (y1[i + 1] * Y_LENGTH / maxRange) - (Y_LENGTH / 2));
317     }
318     popMatrix();
319 }
320
321 /* Arduino からのデータ受信時の処理 */
322 void receive( byte[] data, String ip, int port ) {
323     String message = new String( data );
324     println( "receive:_"+""+message+"\"_from_"+"ip"+"_on_"port_"+"port );
325     if (port == 5556){
326         String dmessage[] = message.split("!", 0);
327         if( dmessage[0].length() != 3 ){
328             String rcvd[] = message.split(",", 0);
329             mil = int(float(rcvd[2]));
330             if (accmilst == 0){
331                 accmilst = mil;
332             }
333             mil = mil - accmilst;
334             fft = float(rcvd[6]);

```

```

335         message = rcvd[0] + "," + rcvd[1] + "," + mil + "," + rcvd
           [3] + "," + rcvd[4] + "," + rcvd[5] + "," + rcvd[6] + "
           ," + rcvd[7] + "," + rcvd[8];
336         output1.println(message);
337     }else{
338         String rcvd[] = message.split(",", 0);
339         mil = int(float(rcvd[2]));
340         if (accmilst == 0){
341             accmilst = mil;
342         }
343         mil = mil - accmilst;
344         acc = float(rcvd[3]);
345         message = rcvd[0] + "," + rcvd[1] + "," + mil + "," + rcvd
           [3] + "," + rcvd[4] + "," + rcvd[5] + "," + rcvd[6] + "
           ," + rcvd[7] + "," + rcvd[8];
346         output1.println(message);
347     }
348 }
349 if (port == 5557){
350     String rcvd[] = message.split(",", 0);
351     sctempmil = int(float(rcvd[2]));
352     if (sctempmilst == 0){
353         sctempmilst = sctempmil - mil - 10;
354     }
355     sctempmil = sctempmil - sctempmilst;
356     sc = float(rcvd[4]);
357     tp = float(rcvd[5]);
358     message = rcvd[0] + "," + rcvd[1] + "," + sctempmil + ",," + sc
           + "," + tp + ",,,";
359     output1.println(message);
360 }
361 }

```

3.3.2 解説

- 記録用パーソナルコンピュータのポートを開き、身体の動きの大きさの測定デバイスと、皮膚コンダクタンスと末梢皮膚温の測定デバイスから送信されたデータを受信し、データの種別を判別し、測定が行われていることを確認するためのリアルタイムグラ

フを表示させ、データを csv ファイルに格納するスケッチである。

- リアルタイムグラフを表示させる部分である、22–25, 33–41, 59–63, 77–319 行は、以下の webpage を参考に記述した。

– <https://garchiving.com/real-time-graph-by-proccesing/>

1–2 行 必要なライブラリの読み込み

4–8 行 皮膚コンダクタンスと末梢皮膚温の測定デバイス及び体の動きの大きさの測定デバイスに書き込まれたスケッチとポート番号を一致させる

10–19 行 データのグラフ表示や csv ファイル格納に必要な変数の宣言

21–28 行 データのグラフ表示, UDP 通信, csv ファイルへの格納に必要なオブジェクトの宣言

30–54 行 グラフ表示, UDP 通信, csv ファイルへの格納に関する設定

56–65 行 グラフ表示の関数を実行

67–75 行 キー操作によってスケッチを終了させる

77–95 行 合成加速度のグラフ表示に関するクラスを定義

97–138 行 合成加速度のデータをグラフに表示させる

140–157 行 周波数のグラフ表示に関するクラスを定義

159–196 行 周波数のデータをグラフに表示させる

198–216 行 末梢皮膚温のグラフ表示に関するクラスを定義

218–260 行 末梢皮膚温のデータをグラフに表示させる

262–279 行 皮膚コンダクタンスのグラフ表示に関するクラスを定義

281–319 行 皮膚コンダクタンスのデータをグラフに表示させる

321–361 行 測定デバイスからデータを受け取った際の処理を行う。325, 349 行目には測定デバイスに書き込まれたスケッチとポート番号を一致させる必要がある。データが送られたポート番号やデータの種類によって処理を分岐させ、グラフ表示のための変数と csv ファイルにデータを格納する

4 測定方法

4.1 測定方法の概要

測定は、皮膚コンダクタンスと末梢皮膚温の測定デバイス、身体の動きの大きさの測定デバイス、記録用パーソナルコンピュータを、インターネットに接続された1台のWi-Fiルーターに接続して行う。皮膚コンダクタンス (μS) は電気抵抗 (Ω) の逆数であるため、デバイス使用の初回に先立って、デバイスの調整と、抵抗値から皮膚コンダクタンスに変換するために必要な値を求め、スケッチに反映して書き込む。

皮膚コンダクタンスの測定方法は通電法である。また、測定デバイスと記録用パーソナルコンピュータとの間で無線通信を行う。そのため、山森他 (2023) では、測定対象の教師、及び授業実施教室内の児童に、心臓ペースメーカーなどの医療機器や、電波によって影響を受ける可能性のある機器（補聴援助システムなど）を利用している者がいないことを確認した。

皮膚コンダクタンス測定の電極と末梢皮膚温測定のセンサを装着する際には、アルコール綿で清拭を行い、サージカルテープで固定した。したがって、測定対象者にアルコールアレルギーや、湿布薬や絆創膏を貼り付けたりした際に皮膚がかぶれたりしたことがないかについても確認した。

4.2 皮膚コンダクタンス測定のための調整

皮膚コンダクタンス測定のための調整は、以下の手順で行う。なお、この調整は一度だけ実施する。測定ごとの調整は行わない。

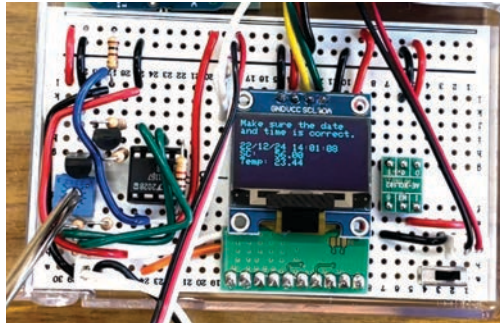
1. 第3.1節に示したスケッチの28-29行目を、以下のように書き換え、マイクロジーメンス変換のための傾きを1、切片を0とする。

```
1 const float slope = 1; //マイクロジーメンス変換のための傾き
2 const float intercept = 0; //マイクロジーメンス変換のための切片
```

2. 書き換えたスケッチを Arduino MKR WiFi 1010 に書き込む。
3. Figure 12 のように、可変抵抗を調整し、皮膚コンダクタンス測定の電極どうしが絶縁の場合に、デバイスのディスプレイに表示される SC の値が 55 前後となるようにする。

Figure 12

皮膚コンダクタンス測定のための調整 (1)



4. Figure 13 のように、 $100\text{k}\Omega$ 、 $1\text{m}\Omega$ の抵抗を用意し、それぞれ、テスターで正確な抵抗値を求め、それぞれの抵抗を皮膚コンダクタンス測定 of 電極にはさみ、表示される値を取得する。

Figure 13

皮膚コンダクタンス測定のための調整 (2)

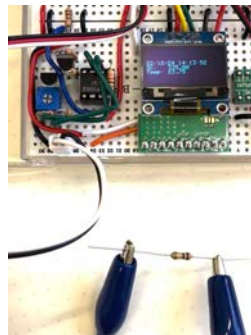
(a) $100\text{k}\Omega$ の正確な抵抗値



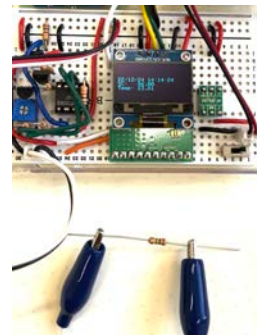
(b) $1\text{m}\Omega$ の正確な抵抗値



(c) $100\text{k}\Omega$ 抵抗をはさんだ場合の SC の表示値



(d) $1\text{m}\Omega$ 抵抗をはさんだ場合の SC の表示値



5. Figure 13 のような場合をまとめると、Table 4 のとおりとなる。したがって、求めたい μS を y 、デバイスに表示される値を x としたとき、 $y = 0.04x - 2.09$ という関係が成り立つ。

Table 4抵抗値, μS , デバイス表示値の関係

抵抗	実際の抵抗値 ($\text{k}\Omega$)	μS	デバイス表示値
100 $\text{k}\Omega$	99.7	10.0	335
1 $\text{m}\Omega$	976	1.0	86

6. このように得た値を, 第 3.1 節に示したスケッチの 28-29 行目に反映させ, Arduino MKR WiFi 1010 に書き込む。

```
1 const float slope = 0.04; //マイクロジーメンス変換のための傾き
2 const float intercept = -2.09; //マイクロジーメンス変換のための切片
```

4.3 皮膚コンダクタンスと末梢皮膚温の測定

皮膚コンダクタンス測定の電極と, 末梢皮膚温測定のセンサの固定と, 測定デバイスの装着は, Figure 14 のとおりに行った。

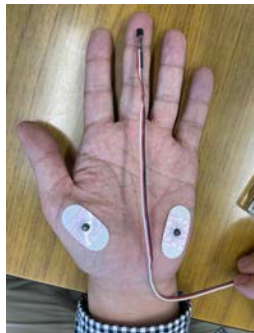
皮膚コンダクタンス測定の電極は, アルコール綿で清拭後, 非利き手の母指球及び小指球に, 使い捨て電極 (日本光電, Vitrode F-150S) を貼り, 剥離を防ぐためにサージカルテープで固定し, ミノムシクリップを接続し, 半指手袋で固定した。

末梢皮膚温測定のセンサは, アルコール綿で清拭後, 非利き手の第三指の末節部にサージカルテープで固定した。これらを測定するデバイスは非利き手の前腕に, マジックベルト (アズワン ナビス 0-1082-13) を用いて固定した。

Figure 14

皮膚コンダクタンス及び末梢皮膚温の測定デバイスの装着

(a) 皮膚コンダクタンス
測定の電極と温度センサ
の位置



(b) 電極とセンサの固定
(1)



(c) 電極とセンサの固定
(2)



(d) デバイスの位置



4.4 身体の動きの大きさの測定

身体の動きの大きさの測定するデバイスは、Figure 15 のとおりに、ネックストラップ付き名札ケース (オープン工業 NX-107) に格納し、ループクリップ (オープン工業 NX-7) を用いて装着した。

Figure 15

身体の動きの大きさを測定するデバイスの装着



4.5 測定データの記録とモニタリング

4.5.1 デバイスの Wi-Fi 接続

Wi-Fi ルーターをインターネットに接続し，皮膚コンダクタンスと末梢皮膚温の測定デバイス，身体の動きの大きさの測定デバイスの電源を入れる。それぞれのデバイスが，以下のような処理を行い，測定が可能となる。

- 第 3.1, 3.2 のスケッチで指定した SSID に接続する。
- NTP サーバーに接続し，現在時刻を取得する。
- 現在時刻と測定値が表示され，測定可能な状態となる。

測定を行う前に，2 台のデバイスが正しい時刻を取得しているかを確認する。取得できていない場合には再起動させる。

4.5.2 記録用パーソナルコンピュータでの Processing のスケッチの起動

記録用パーソナルコンピュータを，皮膚コンダクタンスと末梢皮膚温の測定デバイス，身体の動きの大きさの測定デバイスが接続している Wi-Fi ルーターに接続し，Processing のスケッチを起動する。皮膚コンダクタンスと末梢皮膚温の測定デバイス，身体の動きの大きさの測定デバイス，これら両方から正しくデータを受け取っている場合には，Figure 16 のような表示が得られる。終了する場合には e キーを押下する。

Figure 16

記録用パーソナルコンピュータでの測定のモニタリング



4.5.3 測定データの保存

測定データは、Processing のスケッチが格納されているディレクトリに、csv 形式のファイルとして保存される。ファイル名は YYYYMMDDhhmmss_log1.csv(Processing のスケッチを起動した日時)となる。データの内容は Table 5 のとおりである。

Figure 17

保存される csv ファイル

```
acc!,22/12/22 09:21:40, 33580, 0.87,,,,0,
acc!,22/12/22 09:21:40, 33600, 0.99,,,,0,
acc!,22/12/22 09:21:40, 33620, 1.12,,,,0,
acc!,22/12/22 09:21:40, 33640, 1.13,,,,0,
acc!,22/12/22 09:21:40, 33660, 1.10,,,,0,
acc!,22/12/22 09:21:41, 33680, 1.16,,,,0,
acc!,22/12/22 09:21:41, 33700, 1.02,,,,0,
acc!,22/12/22 09:21:41, 33720, 0.92,,,,0,
acc!,22/12/22 09:21:41, 33740, 1.23,,,,0,
acc!,22/12/22 09:21:41, 33760, 1.22,,,,0,
acc!,22/12/22 09:21:41, 33780, 1.00,,,,0,
acc!,22/12/22 09:21:41, 33800, 0.63,,,,0,
SCtemp!,22/12/22 09:21:41,405242,,22.351,19.0,,,
aFFT!,22/12/22 09:21:41,23579,,,1.95, 50,256
acc!,22/12/22 09:21:41, 33978, 1.42,,,,0,
acc!,22/12/22 09:21:41, 33998, 0.79,,,,0,
acc!,22/12/22 09:21:41, 34018, 0.74,,,,0,
acc!,22/12/22 09:21:41, 34038, 0.91,,,,0,
```

Table 5

csv ファイルの内容

列	内容
1	データの種類
2	日付時刻
3	測定デバイス起動からの経過時間 (ms)
4	合成加速度
5	皮膚コンダクタンス
6	末梢皮膚温
7	身体の動きの周波数
8	サンプリング周波数
9	データをそろえるためのダミーデータ

5 測定データの取り出しと可視化

5.1 測定データの取り出し

記録用パーソナルコンピュータに格納された測定データは、分析対象時間帯以外のデータが含まれている。また、皮膚コンダクタンスと末梢皮膚温のデータと、身体の動きの周波数データは、異なる行に格納されている。分析を行うには、分析対象時間帯に限定した上で、皮膚コンダクタンスと末梢皮膚温のデータと、身体の動きの周波数のデータとを分ける必要がある。

分析対象時間帯に限定した皮膚コンダクタンスと末梢皮膚温のデータと、身体の動きの周波数のデータを、測定データから取り出すための R のコードを示すと以下のとおりである。なお、以下の例では、2022 年 12 月 25 日 10 時 28 分 00 秒から 10 分間のデータを取り出した。

```
1 # 測定データの読み込み
2 setwd("/XXX/XXX")
3 rawdata <- read.csv("20221225102717_log1.csv", header=F)
4 nonadata <- rawdata[!is.na(rawdata[,5]),]
5 #分析対象の列を取り出す
6 data <- nonadata[,2:6]
7 #列名を振る
8 colnames(data)<-c("dt", "et", "acc", "sc", "temp")
9
10 # 分析対象時間帯の指定
11 ## 開始時刻をYYYY/MM/DD hh:mm:ss で入力する
12 start <- grep(pattern = "22/12/25_10:28:00", x = data[,1])[1]
13 ## 終了時刻をYYYY/MM/DD hh:mm:ss で入力する
14 end_ <- grep(pattern = "22/12/25_10:38:00", x = data[,1])
15 ## 分析対象データの最後を指定
16 end <- end_[length(end_)]
17
18 # 皮膚コンダクタンスと末梢皮膚温の分析用データ取り出し
19 ## データの整理
20 ### 日付と時間のデータ
21 dt <- data[start:end, 1]
22 ### 経過時間データ(ミリ秒から分に変換)
23 et <- (data[start:end, 2] - data[start, 2]) / 1000 / 60
```

```

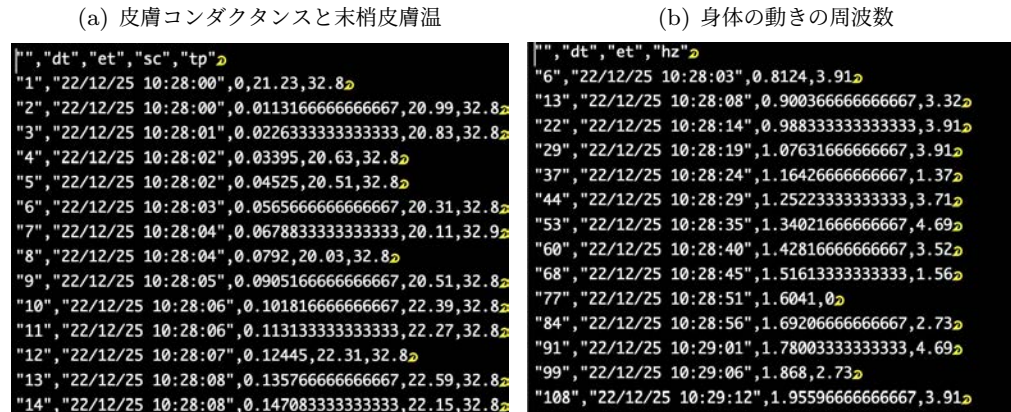
24 ### 皮膚コンダクタンスのデータ
25 sc <- data[start:end, 4]
26 ### 末梢皮膚温のデータ
27 tp <- data[start:end, 5]
28 ### データフレームに変換して結合する
29 dt <- data.frame(dt); et <- data.frame(et)
30 sc <- data.frame(sc); tp <- data.frame(tp)
31 sctp <- cbind(dt, et, sc, tp)
32
33 ## csv 形式で保存
34 setwd("/XXX/YYYY")
35 write.csv(sctp, "sctp.csv")
36
37 # 身体の動きの周波数の分析用データ取り出し
38 hzdata_ <- rawdata[c(2, 3, 7)]
39 hzdata <- hzdata_[!is.na(hzdata_[,3]),]
40 colnames(hzdata) <- c("dt", "et", "hz")
41 library(dplyr)
42 lthz <- left_join(dt, hzdata, by = "dt")
43 lthz <- na.omit(lthz)
44 lthz <- unique(lthz)
45 lthz$et <- (lthz[,2] - start) / 1000 / 60
46 ## csv 形式で保存
47 write.csv(lthz, "lthz.csv")

```

以上のコードを実行し、皮膚コンダクタンスと末梢皮膚温の分析用データと、身体の動きの周波数の分析用データを取り出した csv ファイルの一部を示すと、Figure 18 のとおりである。

Figure 18

取り出した分析用データの例



5.2 皮膚コンダクタンスと末梢皮膚温の変化の可視化

山森他 (2023) で示したような、皮膚コンダクタンス変化と末梢皮膚温の変動のプロットを得るための R のコードを示すと以下のとおりである。一般的な授業時間である 45 分を超える程度の範囲を想定しているため、経過時間が 51 分までとなっているが、ここでは 10 分間のサンプルデータを可視化した。

```
1 # PDF の保存先
2 setwd("/XXX/YYYY")
3 # 末梢皮膚温の原点を決めて入力する
4 temp.min <- 24
5 # 皮膚コンダクタンスの範囲を決めて入力する
6 srange <- c(0, 50)
7
8 # Mactintosh の場合
9 quartz(type = "pdf", file = ("plot.pdf"), width = 12, height = 4)
10 par(family = "HiraKakuProN-W3")
11 # Windows の場合
12 #pdf("plot.pdf", width = 12, height = 3)
13 #par(family="Japan1GothicBBB")
14
15 # マージンをとる
16 par(mar=c(4,5,4,4))
```

```

17 # 皮膚コンダクタンスのプロット
18 plot(sctp$et,sctp$sc,
19      type="l",
20      xlab="経過時間 (分)",
21      cex.lab = 1.0,
22      cex.main = 1.0,
23      ylim = srange,
24      xlim = c(0, 51),
25      ylab = "皮膚コンダクタンス $\mu$  (s)",
26      font.main = 1,
27      pch = 20,
28      cex = 0.1,
29      col = 2,
30      axes = F)
31 # 末梢皮膚温のプロット
32 points(sctp$et, (sctp$tp - temp.min) * 50 / 15,
33 ## - temp.min) * 50 / 15 は重ね描きのための調整
34      type = "l",
35      pch = 20,
36      cex = 0.1,
37      col = 4)
38
39 # 皮膚コンダクタンスの軸
40 axis(side = 2,
41      at = c(0, 10, 20, 30, 40, 50),
42      labels = c(0, 10, 20, 30, 40, 50),
43      cex.axis = 1.0)
44
45 # 末梢皮膚温の軸
46 axis(side = 4,
47      at = c(0, 10, 20, 30, 40, 50),
48      labels = c(23, 26, 29, 32, 35, 38),
49      cex.axis = 1.0, # 追加
50      )
51 mtext(text="末梢皮膚温 $^{\circ}$  (C)",
52      side = 4,
53      cex = 1.0,
54      padj = 4)
55

```

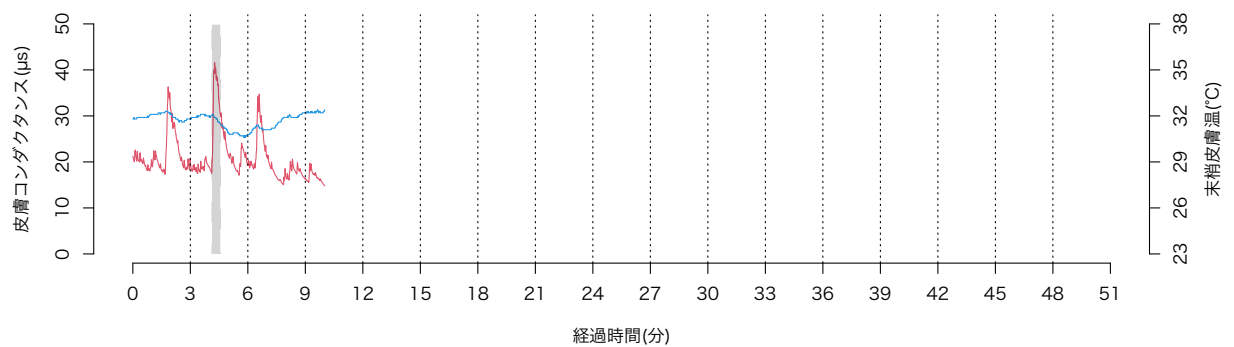
```

56 # 経過時間の軸
57 axis(side=1,
58       at=c(0, 3, 6, 9, 12, 15, 18, 21, 24,
59           27, 30, 33, 36, 39, 42, 45, 48, 51),
60       labels = c(0, 3, 6, 9, 12, 15, 18, 21, 24,
61                27, 30, 33, 36, 39, 42, 45, 48, 51),
62       cex.axis = 1.0)
63
64 # 3分ごとに縦線を入れる
65 abline(v = c(3, 6, 9, 12, 15, 18, 21, 24,
66            27, 30, 33, 36, 39, 42, 45, 48), lty=3)
67
68 # 聞き取り時間帯の網掛け
69 ## 以下の例では4.1分から4.6分の間に網掛けをしている
70 rect(4.1, 0, 4.6, 50, col = rgb(0, 0, 0, alpha = 0.1), lty = 0)

```

このコードを用いて出力したプロットを示すと、Figure 19 のとおりである。

Figure 19
プロットの例



引用文献

- 美濃哲郎 (1986). 皮膚コンダクタンス水準と皮膚コンダクタンス反応 新美良純・鈴木二郎 (編) 皮膚電気活動 (pp. 39-43.) 星和書店.
- 山森光陽・長野祐一郎・徳岡 大・草薙邦広・大内善広 (2023). 生理心理学的指標を用いた授業中の教師の認知負荷の把握 日本教育工学会論文誌, 47(1), 127-139.
<https://doi.org/10.15077/jjet.46054>