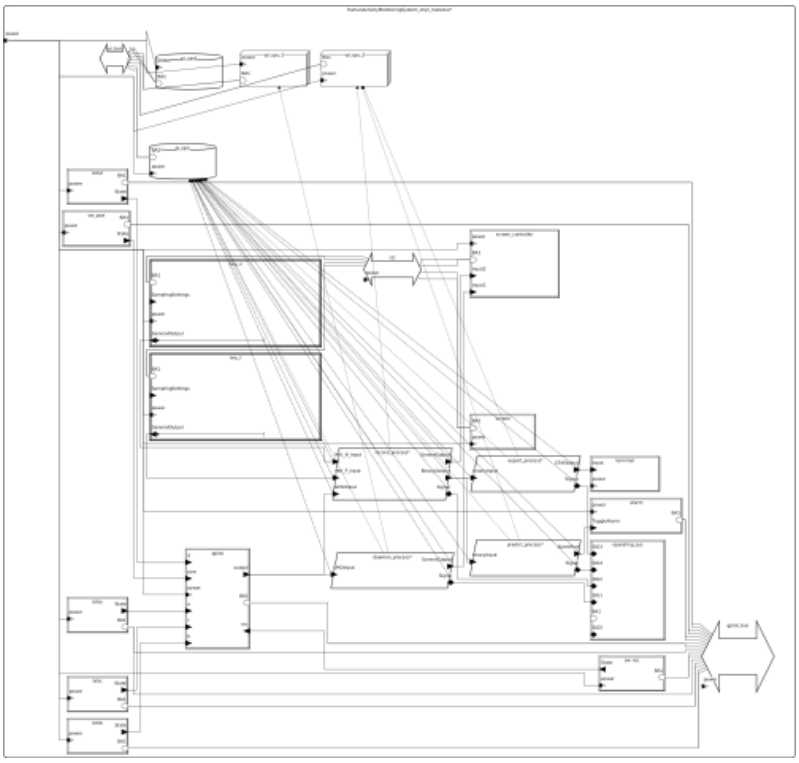
System monitorowania aktywności człowieka - projekt półformalny z wykorzystaniem języka modelowania AADL

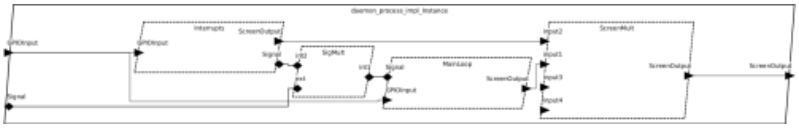
1. System opisuje urządzenie noszone przez człowieka i jego oprogramowanie wnioskujące na podstawie wskazań IMU (akcelerometrów i żyroskopów) o typie wykonywanej czynności, w tym o ewentualnych upadkach. Jest dość wiernym odwzorowaniem istniejącego projektu wykonanego na przedmiocie "inteligentne systemy sensoryczne".
2. Komponenty (łącznie 24, nie licząc wątków wewnątrz procesów):
   - Mikrokontroler/komputer (Raspberry PI)
     - 4 procesy
       (a) `record_process` - proces rejestrujący dane, składający się z następujących wątków:
           i. `MainLoop` - główna pętla, odczytująca czujniki po I2C z częstotliwością ok. 25Hz.
           ii. `Interrupts` - obsługuje sprzętowe przerwania na pinach GPIO (General Purpose Input/Output) oznaczające zmianę stanów przycisków oraz sygnały przesyłane przez system operacyjny (w szczególności wygenerowane przez demona).
           iii. `ScreenWriter` - pisanie na ekran jest czasochłonne i nie powinno przeszkadzać w pomiarach, dlatego umieszczone jest w osobnym wątku.
       (b) `export_process` - proces eksportujący dane (np. na terminal), jednowątkowy.
       (c) `predict_process` - proces realizujący przewidywanie semantyki zachowania człowieka na podstawie obserwabli (jednowątkowy).
       (d) `daemon_process` - daemon zarządzający rejestratorem i pozostałymi procesami (uruchamiając je lub zabijając w zależności od oczekiwanego stanu systemu). W szczególności pilnuje, by proces nagrywający pozostawał żywy, gdy przełącznik `sw_rec` jest załączony, uruchamiając go ponownie w razie potrzeby, a także diagnozuje ewentualne awarie sprzętowe (rozłączenie magistrali I2C) i informuje o nich.
           i. `MainLoop` - wątek zawierający logikę periodyczną (kontrola procesów podległych i sprzętu, wyświetlanie zegarka).
           ii. `Interrupts` - wątek zawierający logikę wyzwalaną w odpowiedzi na zdarzenia, w szczególności sygnały systemowe oraz ze sprzętowych przełączników.
     - Urządzenia reprezentujące komputer
       * `pi_cpu_1`, `pi_cpu_2` - reprezentują dwa rdzenie maszyny cyfrowej
       * `pi_ram` - pamięć operacyjna (w rzeczywistości 512MB)
       * `pi_card` - karta micro SD, jedyna pamięć trwała maszyny
       * `pi_bus` - wewnętrzna magistrala komputera

        * `i2c` - magistrala w standardzie I2C, do komunikacji z sensorami i ekranem
        * `gpios_bus` - reprezentuje zbiorczo bezpośrednie połączenia przycisków do pinów Raspberry
        * `operating_sys` - pseudourządzenie, modelujące funkcjonalność systemu POSIXowego odbierającą i przesyłającą sygnały pomiędzy procesami, używaną szczególnie przez demona.
    - `imu_n`, `imu_f` - 2x IMU - Inertial Measurement Unit - (Sparkfun 9DOF Sensor Stick)- trzyosiowy akcelerometr, żyroskop i magnetometr.
    - Ekran LCD
        - `screen` - odpowiada fizycznemu monochromatycznemu ekranowi OLED
        - `screen_controller` - kontroler ekranu SSD1306 na I2C
    - Przyciski i przełączniki (x6)
        - `sw_pow` (`ToggleSwitch`) - jego przesunięcie w pozycje wyłączenia powoduje wydanie komendy 'shutdown now' prez demona, o ile nagrywanie nie jest w toku
        - `sw_rec` (`ToggleSwitch`) - gdy jest w pozycji załączonej, demon pilnuje by nagrywanie trwało.
        - Cztery przyciski monostabilne, potrzebne by ręcznie etykietować nagrywane dane rodzajem czynności, by umożliwić uczenie nadzorowane na nich.
            * `brba` (`PushButton`) - Big Red Button A
            * `brbb` (`PushButton`) - Big Red Button B
            * `brbc` (`PushButton`) - Big Red Button C
            * `brbd` (`PushButton`) - Big Red Button D
    - Urządzenia wyjściowe danych
        - `terminal` - oprócz przycisków, komunikacja z systemem odbywa się za pomocą terminala. Wydanie odpowiedniego polecenia uruchamia proces eksportu, czyli zwyczajnie wypisania na końcówkę nagromadzonych do tej pory (bądź aktualnie zbieranych) danych. Należy zauważyć, że rzeczywiste urządzenie, działając pod systemem uniksowym, może obsługiwać naraz wiele terminali, w tym wirtualne ssh, pozwalające na przesyłanie danych poprzez sieć. Jednakże część systemu po stronie "klienta", odbierająca te dane, ani rzeczywista możliwość równoczesnego działania kilku instancji procesu `export_process` nie została uwzględniona w tym modelu, jako że nie wpływa zasadniczo na jego strukturę i wynika właściwie z funkcjonalności zapewnianej przez sam system operacyjny (wielodostęp do plików, wirtualne terminale, ssh itd.).
    - System alarmowy
        - `alarm` (`AlarmSystem`) - pozostał jedynie na etapie projektu w rzeczywistej jego implementacji; miał za zadanie ostrzegać przede wszystkim o upadkach, czy to poprzez sygnał dźwiękowy, czy wysyłanie powiadomień na zarejestrowane w systemie telefony.
3. Model - diagramy

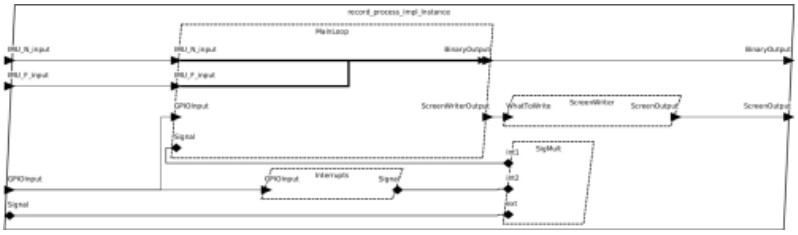
(a)



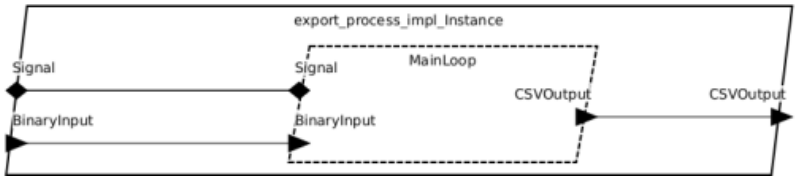
(b)

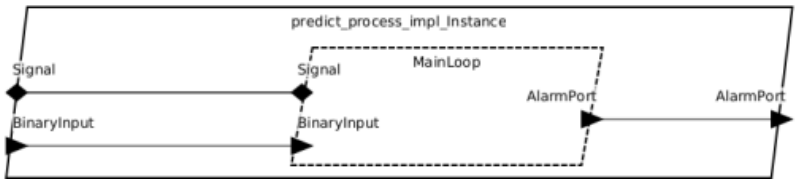

daemon_process_impl_instance
GPIOInput
Interrupts
ScreenOutput
SigMult
MainLoop
ScreenMult
Signal

(c)


record_process_impl_instance
MainLoop
IMU_N_input
IMU_F_input
BinaryOutput
GPIOInput
ScreenWriterOutput
WhatToWrite
ScreenWriter
ScreenOutput
Signal
SigMult
Interrupts

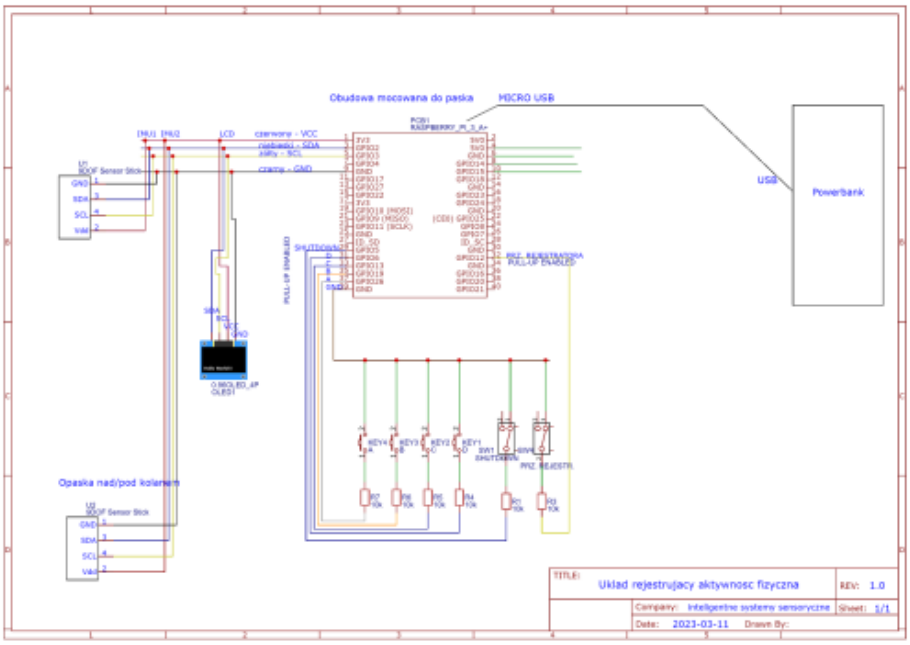
(d)

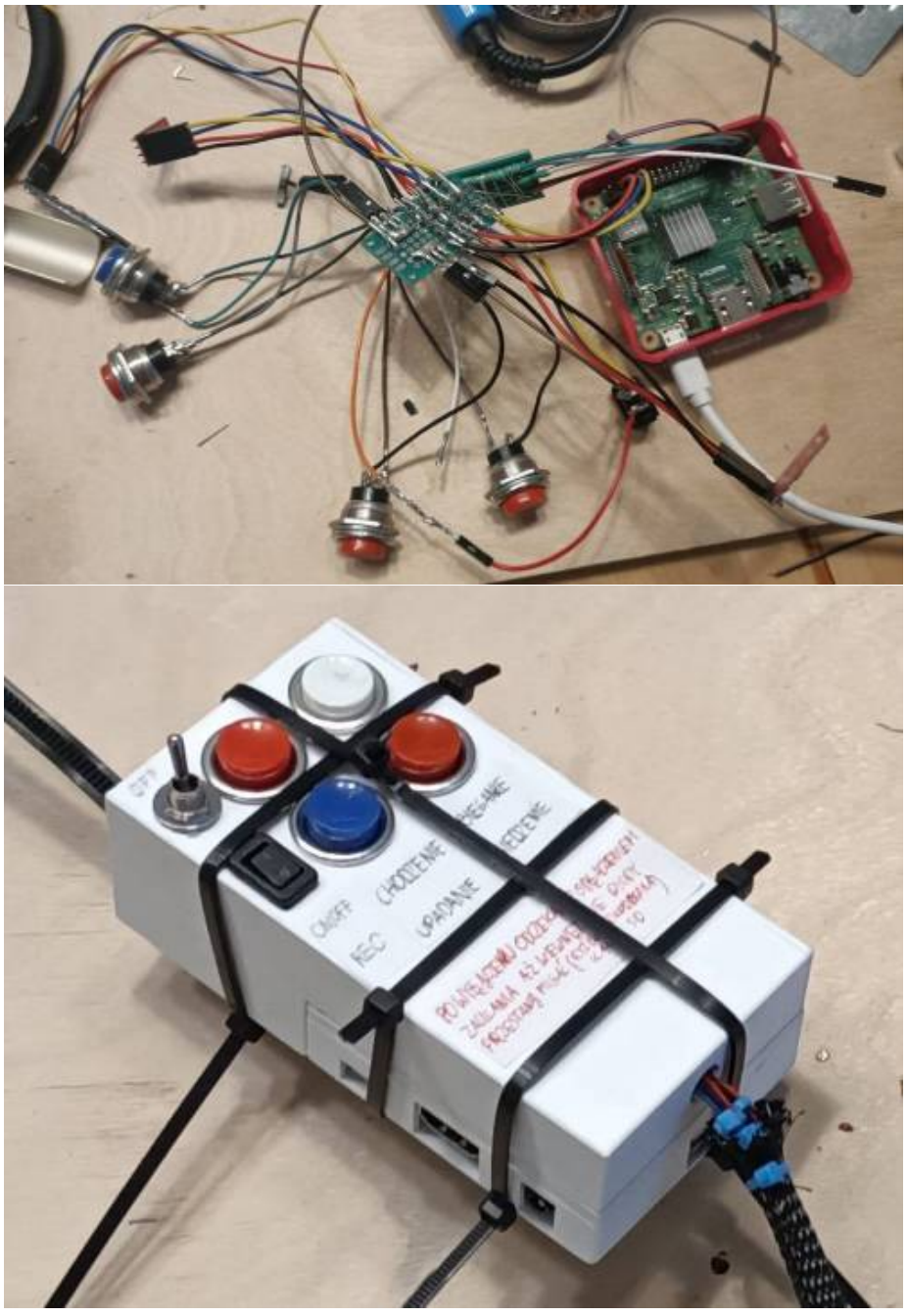

export_process_impl_instance
Signal
Signal
MainLoop
CSVOutput
CSVOutput
BinaryInput
BinaryInput



(e)

4. Fizyczna realizacja projektu:





NAGRYWANIE WYŁĄCZNONE

nic się nie dzieje

urządzenie niedostępne

pod przestawieniu wyłącznika, odczekać 10 sek.

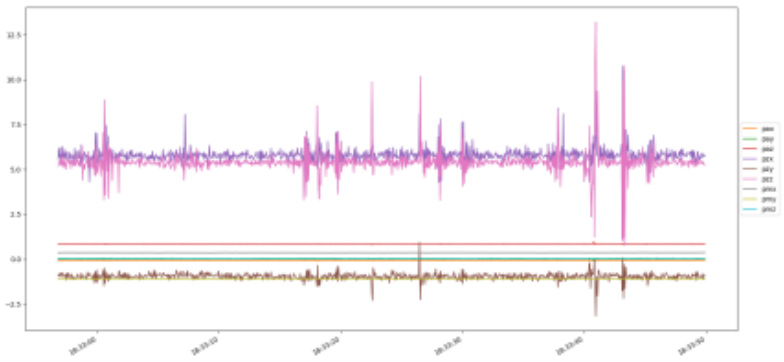
NAGRYWANIE WŁĄCZONE

ETYKIETA

CZAS NAGRYWANIA (MIN:SEK)

SZACOWANA DŁUGOŚĆ CAŁEGO ZAPISU W MIN.

opcjonalny wykrzyknik przed cyfrą oznacza, że z powodu niemożności otworzenia pliku dane/zapis, nagranie znajduje się w innym pliku (dane/zapis_[0-9]+) (uwaga przy imporcie)



Projekt w języku AADL

```
1
2   package smacz
3   public
4           with data_model;
5           with Base_Types;
6
7           data RawIMUVec1
8                   properties
9                           Data_Model::Data_Representation => Float;
10          end RawIMUVec1;
11
12          data RawIMUVec3
13          end RawIMUVec3;
14
15          data implementation RawIMUVec3.Cartesian
16                  subcomponents
17                  x : data RawIMUVec1;
18                  y : data RawIMUVec1;
19                  z : data RawIMUVec1;
20    end RawIMUVec3.Cartesian;
21
22    data imu_out_type
23    end imu_out_type;
24
25          data implementation imu_out_type.impl
26                  subcomponents
27                  Accelerometer : data RawIMUVec3;
28                  Gyroscope : data RawIMUVec3;
29                  Magnetometer : data RawIMUVec3;
30    end imu_out_type.impl;
31
32          data IMUSamplingSettingsType
33          end IMUSamplingSettingsType;
34
35          device IMU
36                  features
37                          AccelerometerOutput: out data port RawIMUVec3;
38                          GyroscopeOutput: out data port RawIMUVec3;
39                          MagnetometerOutput: out data port RawIMUVec3;
40                          GeneralOutput: out data port imu_out_type;
41                          SamplingSettings: in data port
                                IMUSamplingSettingsType;
42                  BA1: requires bus access i2c_bus;
43          end IMU;
44
45          device implementation IMU.Near
46          end IMU.Near;
47          device implementation IMU.Far
48          end IMU.Far;
49
50          device PhysicalLCDScreen
51                  features
52                      BA1: requires bus access i2c_bus;
53          end PhysicalLCDScreen;
54
55          data screen_command_type
56          end screen_command_type;
57
58          device ScreenController
59                  features
60                          BA1: requires bus access i2c_bus;
61                          input1: in data port screen_command_type;
62                          input2: in data port screen_command_type;
63          end ScreenController;
64
65          device AlarmSystem
66                  features
```

```
67                      ToggleAlarm: in data port Base_Types::Boolean;
68                      BA1: requires bus access gpio_bus;
69      end AlarmSystem;
70
71      device PushButton
72              features
73                      State: out data port Base_Types::Boolean;
74                      BA1: requires bus access gpio_bus;
75      end PushButton;
76
77      device ToggleSwitch
78              features
79                      State: out data port Base_Types::Boolean;
80                      BA1: requires bus access gpio_bus;
81      end ToggleSwitch;
82
83      data gpios_type
84      end gpios_type;
85
86      data implementation gpios_type.impl
87              subcomponents
88              a : data Base_Types::Boolean;
89              b : data Base_Types::Boolean;
90              c : data Base_Types::Boolean;
91              d : data Base_Types::Boolean;
92              rec : data Base_Types::Boolean;
93              pow : data Base_Types::Boolean;
94      end gpios_type.impl;
95
96      device GPIO_Aggregator
97              features
98                      output: out data port gpios_type;
99                      a: in data port Base_Types::Boolean;
100                     b: in data port Base_Types::Boolean;
101                     c: in data port Base_Types::Boolean;
102                     d: in data port Base_Types::Boolean;
103                     rec: in data port Base_Types::Boolean;
104                     pow: in data port Base_Types::Boolean;
105
106                     BA1: requires bus access gpio_bus;
107     end GPIO_Aggregator;
108
109     data signal_type
110     end signal_type;
111
112     device Terminal
113             features
114                     Input: in data port Base_Types::String;
115     end Terminal;
116
117
118     bus internal_bus
119     end internal_bus;
120
121     bus implementation internal_bus.impl
122     end internal_bus.impl;
123
124     bus i2c_bus
125     end i2c_bus;
126
127     bus implementation i2c_bus.impl
128     end i2c_bus.impl;
129
130     bus gpio_bus
131     end gpio_bus;
132
133     bus implementation gpio_bus.impl
134     end gpio_bus.impl;
```

```
135
136         processor ARMv8
137         features
138         BA1: requires bus access internal_bus;
139         end ARMv8;
140
141         memory RAM
142                 features
143                 BA1: requires bus access internal_bus;
144         end RAM;
145
146         memory implementation RAM.impl
147         end RAM.impl;
148
149         memory SD_card
150                 features
151                 BA1: requires bus access internal_bus;
152         end SD_card;
153
154         memory implementation SD_card.micro32GB
155         end SD_card.micro32GB;
156
157 ------ SUPERVISING DAEMON ------
158
159         thread DaemonMainLoop
160                 features
161                         GPIOInput: in data port gpios_type;
162                         ScreenOutput: out data port screen_command_type;
163                         Signal: in out data port signal_type;--for
                                killing and maybe dummy for creating
                                processes
164                 properties
165                         Dispatch_Protocol => Periodic;
166                         Dispatch_Offset => 4ms;
167                         Deadline => 150ms;
168                         Period => 1000ms;
169                         Compute_Execution_Time => 0ms .. 50ms;
170         end DaemonMainLoop;
171         thread implementation DaemonMainLoop.impl
172         end DaemonMainLoop.impl;
173
174         thread DeamonInterrupts
175                 features
176                         GPIOInput: in data port gpios_type;
177                         Signal: in out data port signal_type;
178                         ScreenOutput: out data port screen_command_type;
179                 properties
180                         Dispatch_Protocol => Sporadic;
181                         Dispatch_Offset => 4ms;
182                         Deadline => 50ms;
183                         Compute_Execution_Time => 0ms .. 30ms;
184         end DeamonInterrupts;
185         thread implementation DeamonInterrupts.impl
186         end DeamonInterrupts.impl;
187
188         process daemon_process
189                 features
190                         GPIOInput: in data port gpios_type;
191                         ScreenOutput: out data port screen_command_type;
192                         Signal: in out data port signal_type;
193         end daemon_process;
194
195         process implementation daemon_process.impl
196                 subcomponents
197                         MainLoop: thread DaemonMainLoop.impl;
198                         Interrupts: thread DeamonInterrupts.impl;
199                 connections
200                         gp_loop: port GPIOInput -> MainLoop.GPIOInput;
```

```
201                     gp_int: port GPIOInput -> Interrupts.GPIOInput;
202                     sig_loop: port Signal <-> MainLoop.Signal;
203                     sig_int: port Signal <-> Interrupts.Signal;
204                     scr_loop: port MainLoop.ScreenOutput ->
                            ScreenOutput;
205                     int_loop: port Interrupts.ScreenOutput ->
                            ScreenOutput;
206         end daemon_process.impl;
207
208         ------ RECORDING MECHANISM ------
209
210         data recording_binary_type
211         end recording_binary_type;
212
213         thread RecordMainLoop
214                 features
215                         IMU_N_input: in data port imu_out_type;
216                         IMU_F_input: in data port imu_out_type;
217                         GPIOInput: in data port gpios_type;
218                         Signal: in out data port signal_type;
219                         ScreenWriterOutput: out data port Base_Types::
                                String;
220                         BinaryOutput: out data port
                                recording_binary_type;
221                 properties
222                         Dispatch_Protocol => Periodic;
223                         Dispatch_Offset => 4ms;
224                         Deadline => 45ms;
225                         Period => 50ms;--20 Hz
226                         Compute_Execution_Time => 0ms .. 30ms;
227         end RecordMainLoop;
228         thread implementation RecordMainLoop.impl
229         end RecordMainLoop.impl;
230
231         thread RecordScreenWriter
232                 features
233                         WhatToWrite: in data port Base_Types::String;
234                         ScreenOutput: out data port screen_command_type;
235                 properties
236                         Dispatch_Protocol => Periodic;
237                         Dispatch_Offset => 6ms;
238                         Deadline => 45ms;
239                         Period => 50ms;--20 Hz
240                         Compute_Execution_Time => 0ms .. 30ms;
241         end RecordScreenWriter;
242         thread implementation RecordScreenWriter.impl
243         end RecordScreenWriter.impl;
244
245         thread RecordInterrupts
246                 features
247                         GPIOInput: in data port gpios_type;
248                         Signal: in out data port signal_type;
249                 properties
250                         Dispatch_Protocol => Sporadic;
251                         Dispatch_Offset => 4ms;
252                         Deadline => 50ms;
253                         Compute_Execution_Time => 0ms .. 30ms;
254         end RecordInterrupts;
255         thread implementation RecordInterrupts.impl
256         end RecordInterrupts.impl;
257
258         process record_process
259                 features
260                         IMU_N_input: in data port imu_out_type;
261                         IMU_F_input: in data port imu_out_type;
262                         GPIOInput: in data port gpios_type;
263                         Signal: in out data port signal_type;
264                         ScreenOutput: out data port screen_command_type;
```

```
265                     BinaryOutput: out data port
                            recording_binary_type;
266         end record_process;
267
268         process implementation record_process.impl
269                 subcomponents
270                         MainLoop: thread RecordMainLoop.impl;
271                         Interrupts: thread RecordInterrupts.impl;
272             ScreenWriter: thread RecordScreenWriter.impl;
273                 connections
274                         gp_loop: port GPIOInput -> MainLoop.GPIOInput;
275                         gp_int: port GPIOInput -> Interrupts.GPIOInput;
276                         sig_loop: port Signal <-> MainLoop.Signal;
277                         sig_int: port Signal <-> Interrupts.Signal;
278                         loop_wri: port MainLoop.ScreenWriterOutput ->
                                ScreenWriter.WhatToWrite;
279                         wri_scr: port ScreenWriter.ScreenOutput ->
                                ScreenOutput;
280                         imu_n_loop: port IMU_N_input -> MainLoop.
                                IMU_N_input;
281                         imu_f_loop: port IMU_F_input -> MainLoop.
                                IMU_F_input;
282                         loop_out: port MainLoop.BinaryOutput ->
                                BinaryOutput;
283         end record_process.impl;
284
285         ------ EXPORTING MECHANISM ------
286
287         thread ExportMainLoop
288                 features
289                         BinaryInput: in data port recording_binary_type;
290                         CSVOutput: out data port Base_Types::String;
291                         Signal: in out data port signal_type;
292                 properties
293                         Dispatch_Protocol => Periodic;
294                         Dispatch_Offset => 4ms;
295                         Deadline => 45ms;
296                         Period => 50ms;
297                         Compute_Execution_Time => 0ms .. 40ms;
298         end ExportMainLoop;
299         thread implementation ExportMainLoop.impl
300         end ExportMainLoop.impl;
301
302         process export_process
303                 features
304                         BinaryInput: in data port recording_binary_type;
305                         CSVOutput: out data port Base_Types::String;
306                         Signal: in out data port signal_type;
307         end export_process;
308
309         process implementation export_process.impl
310         subcomponents
311                         MainLoop: thread ExportMainLoop.impl;
312                 connections
313                         loop_bin: port BinaryInput -> MainLoop.
                                BinaryInput;
314                         loop_csv: port MainLoop.CSVOutput -> CSVOutput;
315                         sig_loop: port Signal <-> MainLoop.Signal;
316         end export_process.impl;
317
318         ------ PREDICTING MECHANISM ------
319
320         thread PedictMainLoop
321                 features
322                         BinaryInput: in data port recording_binary_type;
323                         AlarmPort: out data port Base_Types::Boolean;
324                         Signal: in out data port signal_type;
325                 properties
```

```
326                         Dispatch_Protocol => Periodic;
327                         Dispatch_Offset => 4ms;
328                         Deadline => 45ms;
329                         Period => 50ms;
330                         Compute_Execution_Time => 0ms .. 40ms;
331         end PedictMainLoop;
332         thread implementation PedictMainLoop.impl
333         end PedictMainLoop.impl;
334
335         process predict_process
336                 features
337                         BinaryInput: in data port recording_binary_type;
338                         AlarmPort: out data port Base_Types::Boolean;
339                         Signal: in out data port signal_type;
340         end predict_process;
341
342         process implementation predict_process.impl
343         subcomponents
344                         MainLoop: thread PedictMainLoop.impl;
345                 connections
346                         loop_bin: port BinaryInput -> MainLoop.
                                BinaryInput;
347                         loop_alm: port MainLoop.AlarmPort -> AlarmPort;
348                         sig_loop: port Signal <-> MainLoop.Signal;
349         end predict_process.impl;
350
351         device SystemAbstraction
352                 features
353                         BA1: requires bus access i2c_bus;
354                         SIG1: in out data port signal_type;
355                         SIG2: in out data port signal_type;
356                         SIG3: in out data port signal_type;
357                         SIG4: in out data port signal_type;
358                         SIG5: in out data port signal_type;
359         end SystemAbstraction;
360
361 --      process ssh_daemon
362 --              features
363 --                      GPIOInput: in data port gpios_type;
364 --                      Signal: in out data port signal_type;
365 --      end ssh_daemon;
366 --
367 --      process implementation ssh_daemon.impl
368 --      end ssh_daemon.impl;
369
370
371         system HumanActivityMonitoringSystem
372         end HumanActivityMonitoringSystem;
373         system implementation HumanActivityMonitoringSystem.impl
374                 subcomponents
375                         daemon_process: process daemon_process.impl;
376                         record_process: process record_process.impl;
377                         export_process: process export_process.impl;
378                         predict_process: process predict_process.impl;
379                         operating_sys: device SystemAbstraction;
380 --                      ssh_daemon: process ssh_daemon.impl;
381                         pi_cpu_1: processor ARMv8;
382                         pi_cpu_2: processor ARMv8;
383                         pi_ram: memory RAM.impl;
384                         pi_card: memory SD_card.micro32GB;
385                         pi_bus: bus internal_bus.impl;
386                         i2c: bus i2c_bus.impl;
387                         gpios_bus: bus gpio_bus.impl;
388                         imu_n: device IMU.Near;
389                         imu_f: device IMU.Far;
390                         screen_controller: device ScreenController;
391                         screen: device PhysicalLCDScreen;
392
```

```
393                         alarm: device AlarmSystem;
394                         brba: device PushButton;--Big Red Button A
395                         brbb: device PushButton;
396                         brbc: device PushButton;
397                         brbd: device PushButton;
398                         sw_pow: device ToggleSwitch;
399                         sw_rec: device ToggleSwitch;
400                         gpios: device GPIO_Aggregator;
401
402                         terminal: device Terminal;
403
404
405                 connections
406                         ----- process connections
407                         --for daemon process
408                         gpios_dae: port gpios.output -> daemon_process.
                                GPIOInput;
409                         dae_scr_contr: port daemon_process.ScreenOutput
                                -> screen_controller.input1;
410                         dae_sys: port daemon_process.Signal ->
                                operating_sys.SIG1;
411                         dae_rec: port record_process.Signal ->
                                operating_sys.SIG2;
412                         dae_exp: port  export_process.Signal ->
                                operating_sys.SIG3;
413                         dae_pred: port predict_process.Signal ->
                                operating_sys.SIG4;
414                         --for recording process
415                         gpios_rec: port gpios.output -> record_process.
                                GPIOInput;
416                         rec_scr_contr: port record_process.ScreenOutput
                                -> screen_controller.input2;
417                         imu_n_rec: port imu_n.GeneralOutput ->
                                record_process.IMU_N_input;
418                         imu_f_rec: port imu_f.GeneralOutput ->
                                record_process.IMU_F_input;
419                         --binary output consumed by exporter and
                                predictor
420                         --rec_dae == dae_rec
421                         --for exporting process
422                         rec_exp: port record_process.BinaryOutput ->
                                export_process.BinaryInput;
423                         exp_term: port export_process.CSVOutput ->
                                terminal.Input;
424                         --for predicting process
425                         rec_pred: port record_process.BinaryOutput ->
                                predict_process.BinaryInput;
426                         pred_alm: port predict_process.AlarmPort ->
                                alarm.ToggleAlarm;
427
428                         ---gpios
429                         a_agr: port brba.State -> gpios.a;
430                         b_agr: port brbb.State -> gpios.b;
431                         c_agr: port brbc.State -> gpios.c;
432                         d_agr: port brbd.State -> gpios.d;
433                         pow_agr: port sw_pow.State -> gpios.pow;
434                         rec_agr: port sw_rec.State -> gpios.rec;
435
436                         -- -- buses
437                         --internal bus
438                         PIBAC1: bus access pi_bus <-> pi_cpu_1.BA1;
439                         PIBAC2: bus access pi_bus <-> pi_cpu_2.BA1;
440                         PIBAC3: bus access pi_bus <-> pi_ram.BA1;
441                         PIBAC4: bus access pi_bus <-> pi_card.BA1;
442                         --i2cbus
443                         I2CAC1: bus access i2c <-> imu_n.BA1;
444                         I2CAC2: bus access i2c <-> imu_f.BA1;
```

```
445                             I2CAC3: bus access i2c <-> screen_controller.BA1
                                    ;
446                             I2CAC4: bus access i2c <-> screen.BA1;
447                             --gpio bus (in reality not a bus...)
448                             GPAC1: bus access gpios_bus <-> gpios.BA1;
449                             GPAC2: bus access gpios_bus <-> brba.BA1;
450                             GPAC3: bus access gpios_bus <-> brbb.BA1;
451                             GPAC4: bus access gpios_bus <-> brbc.BA1;
452                             GPAC5: bus access gpios_bus <-> brbd.BA1;
453                             GPAC6: bus access gpios_bus <-> sw_pow.BA1;
454                             GPAC7: bus access gpios_bus <-> sw_rec.BA1;
455                             GPAC8: bus access gpios_bus <-> alarm.BA1;
456                             --
457                             --temp_conn :port tempsensor.Temp ->
                                    control_process.TempInput;
458                             -- pass
459                             --abstrakcyjne komponenty w aadlu
460
461                     properties
462                             Actual_Processor_Binding => (reference(pi_cpu_1)
                                    ) applies to daemon_process;
463                             Actual_Memory_Binding => (reference(pi_ram))
                                    applies to daemon_process;
464                             Actual_Processor_Binding => (reference(pi_cpu_2)
                                    ) applies to record_process;
465                             Actual_Memory_Binding => (reference(pi_ram))
                                    applies to record_process;
466                             Actual_Processor_Binding => (reference(pi_cpu_2)
                                    ) applies to export_process;
467                             Actual_Memory_Binding => (reference(pi_ram))
                                    applies to export_process;
468                             Actual_Processor_Binding => (reference(pi_cpu_2)
                                    ) applies to predict_process;
469                             Actual_Memory_Binding => (reference(pi_ram))
                                    applies to predict_process;
470                             --Actual_Memory_Binding => (reference(pi_card))
                                    applies to daemon_process;
471             end HumanActivityMonitoringSystem.impl;
472
473
474     end smacz;
```