Projektbericht 5. Semester

André Niemann - 2009.... und Martin Miedreich - 20090032

25. Februar 2012

Inhaltsverzeichnis

1 Organisation

Die allgemeine Kommunikation war gut, die meist wöchentlichen Meetings zum feststellen des Fortschrittes und absprechen der nächsten schritte ebenso. Unterstützend könnte man dazu Projektverwaltungssoftware wie Trac oder Redmine einsetzen, bei der auch Arbeitspakete erstellt, gewissen Personen als Bearbeiter und anderen als Hypervisor zugewiesen werden können. Damit kann man den Fortschritt auch gut verfolgen,wenn man sich mal nicht trifft und man selbst sieht auch auf einem Blick wo man steht.

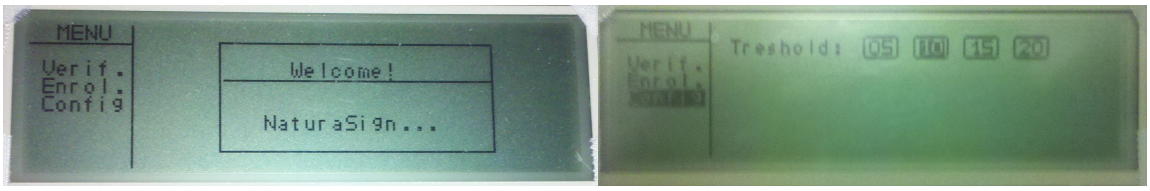
Zudem hat man durch die Notizen zur den Arbeitspaketen auch gleiche eine Art Doku, wodurch sich der Aufwand dafür minimieren würde. Eine Intensivere nutzung von SCM, bzw erstmal ein zuverlässiges und Zeitgemäßes SCM wär auch wünschenswert. In Projekten mit mehreren Entwicklern, die auch verteilt arbeiten sollte sowas heutzutage zum allgemeinen Arbeitsfluss gehören. Die vorteile dafür sind gemäß der Device "commit early and often"man selbst und andere sehen sofort was als letztes verändert wurde und in einem kurzen knackigen Kommentar für die Log auch ohne in die Source zu gucken. man kann das ganze auch ein Bugtrackingsystem koppeln (siehe trac/redmine) und kann Bugs fillen und zu den commits zuordnen. Es kann auch ein Buildserver angebunden werden, der je nach policy z.b. commitgesteuert Testet (siehe TDD) oder einfach so versucht das Projekt zu bauen und bei einem Fehler wird sofort Alarm geschlagen um regressionen zu vermeiden.

Eine zusätzliche Verbesserung wäre es die Quelltexte durch eine Dokutool (z.b Doxygen) zu jagen um damit eine dedizierte Dokumentation über die Funktionen, quasi der API, zu haben. So kann man auch mal außerhalb der Entwicklungsumgebung sich gedanken zum Programmablauf machen.

2 Implementierung

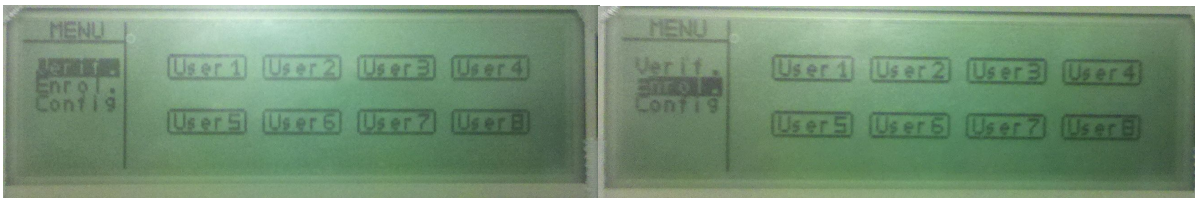
Die Implementierung gestalltete sich überwiegend in der main.c. Da diese mit am schlechtesten auskommentiert war, war eine Rekapitulaion des bereits vorhandenen Quellcodes sehr schwierig. Wir entschlossen uns daher die main.c schrittweise neu herzuleiten und die Verwendung der einzelnen Headerdateien zu verstehen. Die ersten GUIs die somit entstanden waren einfache Konstrukte aus Linien und Text. Im nächsten Schritt implementierten wir die ersten Buttons und die dazugehörigen Funktionen. Aufbauend darauf enstand ein simpler Taschenrechner für dessen Eingabe ein 3x4 Pinpad und jeweils ein Button für die jeweilige Grundrechenart. Wir kamen danach zu dem Punkt an dem wir uns ein Konzept für die eigentliche Nutzeroberfläche überlegen mussten. Dabei entschieden wir uns für die Menüführung welche bereits in der orgninalen main.c implementiert war, da diese durch die Menüauswahl auf der rechten Displayseite eine intuitive Bedienung ermöglichte. Die Menüpunkte ermöglichen dem Nutzer Zugriff auf die Enrollement-, Verifikations- u. Konfigurationsfunktionen zu nehmen. Nach der Initialisierung wird auf dem Display eine Willkommensanzeige dargestellt. (Abbildung 1a) Beim Enrollement und der Verifikation werden jeweils 8 Buttons angezeigt welche für die Auswahl des aktuellen Nutzers zustädnig sind. Die Buttons wurden einem Frame in der Anordung 4x2 zugeordnet, damit mussten wir die Buttons nicht einzeln zeichnen lassen sondern konnten durch ein einmaliges zeichnen des Frames alle Buttons darstellen. (Abbildung 2) Um den Quellcode weiterhin zu verkürzen nutzten wir den Frame mit den Buttons wie bereits erwähnt sowohl für den Enrollementprozess sowie für die Verifikation. Das heißt bei einem Klickevent wird die Abfrage gestellt welche Menüpunkt aktiviert ist und erst danach erfolgt der Aufruf der Funktion. Diese Funktion ist bisweilen aber nur für User1 implementiert. Um die Tresholdeinstellung aus dem Konfigurationsmenü in die Verifikation mit einfließen zu lassen haben wir die Funktion dafür geringfügig modifiziert. Wir haben einen zusätzlichen übergabeparameter und eine Abfrage ob die Hammingdistanz den gesetzten Treshold überschritten hat hinzugefügt. Ausserdem erfolgt in der Funktion noch die Ausgabe ob die Verifikation erfolgreich war oder eben nicht.(Abbildung 3) Das zweite Frame ist demzufolge für das Konfigurationsmenü geschaffen. Dort besteht die Möglichkeit den Treshold für eine gültige bzw. ungültige Verifikation auf die Werte 5, 10, 15 und 20 zu setzten (Abbildung 1b). Sollte hier keine weitere Einstellung des Nutzers erfolgen wird die Verifikation mit einem Treshold von 10 durchgeführt. [1]

[1] Quellcode: 4



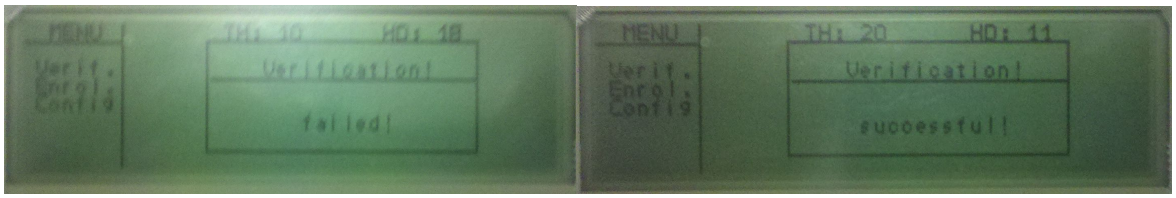
(a) (b)

Abbildung 1: Willkommensanzeige und Konfigurationsmenü



(a) (b)

Abbildung 2: Nutzerauswahl beim Enrollment und der Verifikation



(a) (b)

Abbildung 3: Ergebnisse der Verifikation

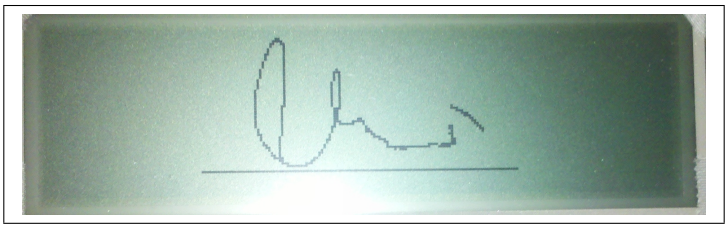


Abbildung 4: Unterschrift

3 Matlab-Vergleiche

Für die Vergleiche der einzelnen Datensätze war es notwendig den vorhandenen Quellcode zu modifizieren. Anfangs stand also primär das Verständnis für den Hashalgorithmus als solchen sowie dessen Implementierung in Matlab im Vordergrund. Dies nahm einige Zeit in Anspruch, da die Umsetzung des Algorithmus zwar mathematisch kompakt aber dadurch schwer verständlich, implementiert worden ist. Als der Quellcode ausreichend modifiziert war fiel uns nach der ersten Versuchsreihe auf, dass die Werte unrealistisch hoch bis hin zu unmöglich waren. Indiz dafür war, dass die Kollisionsrate und die Reproduktionsrate jedesmal bei 0 lag. Dies hätte eine Verifizierung unmöglich gemacht. Die Ursache dafür war letztendlich, dass für die Simulation ein Toleranzvektor mit dem Wert 0 verwendet wurde. Um also realistischere Werte zu erzielen setzten wir diesen auf 1 mit dem Ergebniss, dass sich die Aussagekraft der Werte deutlich besserte. Nach der ersten Versuchsreihe fiel uns auf, dass die Ergebnis unrealistisch bis hin zu unmöglich waren. Indiz dafür war dass die CR und die RR jedesmal bei 0 lag. Was eine Verifizierung unmöglich gemacht hätte. Ebenfalls war der Treshold zu hoch. Uns fiel uns dann auf, dass die Ursache hierfür der Toleranzvektor war. Dieser war in der bisherigen Umsetzung immer auf 0. Für das weitere Testverfahren nutzen wir dann einen Toleranzvektor von 1. Anschließend hatten die Ergebnisse auch eine wesentlich bessere Aussagekraft. Die besten Ergebnisse lieferten die Daten welche zeitnah aufgenommen und auch verifiziert wurden. Die Durchschnittswerte betrugen dabei:

$$EER : 0{,}027950\ Treshold : 13{,}0561\ RR : 0{,}064906$$

Diese sind deutlich geringer als der Vergleich der Samples welche einen zeitlichen Abstand von 1-2 Monaten haben. Hierbei liegen die Durchschnittswerte bei:

$$EER : 0{,}104641\ Treshold : 19{,}8423\ RR : 0{,}010566$$

Ähnliche Werte wurden auch beim Vergleich jüngerer Referenzdaten mit älteren Verifikationsdaten (Tabelle 1), älterer Referenzdaten mit neueren Referenzdaten (Tabelle 2) und älterer Verifikationsdaten mit neueren Verifikationsdaten (Tabelle 3). Zwischen der Aufnahme dereinzelnen Datensätze lag jeweils ein Zeitraum von einem Monat. Basierend auf diesen Daten haben wir dann bei der Implementation auf dem Gerät einen maximalen Treshold von 20 gewählt.

4 Quellcode

4.1 main.c

```
1  //////////////////////////////////////////////////////////////////////
   // $Id: main.c 155 2009-10-23 13:10:11Z student $
3  //
   // $Author: student $
5  // Copyright by Simple Solutions, Stephan Schirrmann, Martin Miedreich,
       Andre Niemann
   //
7  // $Project$
   //
9  // File Description:
   //
11 //////////////////////////////////////////////////////////////////////

13 #include <LPC23xx.H>
   #include <stdio.h>
15 #include "type.h"
   #include "config.h"
17 #include "hiduser.h"
   #include "ports.h"
19 #include "touch.h"
   #include "spi.h"
21 #include "display.h"
   #include "rtc_nvram.h"
23 #include "timer.h"
   #include "serial.h"
25 #include "usbcmd.h"
   #include "pll.h"
27 #include "ethernet.h"
   #include "flash.h"
29 #include "dc.h"
   #include "gui.h"
```

Tabelle 1: jüngere Referenzdaten vs ältere Verifikationsdaten

R2 vs. V1	EER	Thr.:	RR	CR	CRR
77993	0,183870	19,5686	0,007547	0	0,496230
pin	0,135180	18,2946	0,007547	0	0,496230
pseudonym	0,109580	23,4906	0,003774	0	0,498110
symbol	0,144660	25,6645	0,003774	0	0,498110
woher	0,093405	24,7493	0,011321	0	0,494340

R3 vs. V1	EER	Thr.:	RR	CR	CRR
77993	0,206380	20,0517	0,000000	0	0,500000
pin	0,193000	20,9760	0,000000	0	0,500000
pseudonym	0,137170	25,3649	0,003774	0	0,498110
symbol	0,156560	23,9185	0,007547	0	0,496230
woher	0,098103	23,6676	0,015094	0	0,492450

R3 vs. V2	EER	Thr.:	RR	CR	CRR
77993	0,121180	15,8864	0,007547	0	0,496230
pin	0,116660	16,6171	0,011321	0	0,494340
pseudonym	0,103180	23,3284	0,022642	0	0,488680
symbol	0,053443	16,4594	0,018868	0	0,490570
woher	0,078964	22,3582	0,022642	0	0,488680

Tabelle 2: ältere Referenzdaten vs neuere Referenzdaten

R1 vs. R2	EER	Thr.:	RR	CR	CRR
77993	0,171200	17,3263	0,003774	0	0,498110
pin	0,128400	17,2437	0,011321	0	0,494340
pseudonym	0,093423	20,8107	0,007547	0	0,496230
symbol	0,074450	19,2118	0,003774	0	0,498110
woher	0,094381	20,4945	0,000000	0	0,500000

R1 vs. R3	EER	Thr.:	RR	CR	CRR
77993	0,227050	18,9436	0,003774	0	0,498110
pin	0,173500	18,8044	0,007547	0	0,496230
pseudonym	0,115290	21,9081	0,011321	0	0,494340
symbol	0,104090	20,5692	0,018868	0	0,490570
woher	0,120010	22,0654	0,015094	0	0,492450

R2 vs. R3	EER	Thr.:	RR	CR	CRR
77993	0,144320	16,3755	0,011321	0	0,494380
pin	0,113010	16,4593	0,003774	0	0,498110
pseudonym	0,077918	21,1759	0,007547	0	0,496230
symbol	0,088011	20,7354	0,030189	0	0,484910
woher	0,060803	22,1775	0,011321	0	0,494340

Tabelle 3: ältere Verifikationsdaten vs neuere Verifikationsdaten

V1 vs. V2	EER	Thr.:	RR	CR	CRR
77993	0,147440	21,6426	0,000000	0	0,500000
pin	0,132110	19,9980	0,007547	0	0,496230
pseudonym	0,089150	25,2292	0,000000	0	0,500000
symbol	0,132170	25,3248	0,026415	0	0,486790
woher	0,072075	24,2250	0,000000	0	0,500000

V1 vs. V3	EER	Thr.:	RR	CR	CRR
77993	0,161760	21,7612	0,000000	0	0,500000
pin	0,161990	20,5092	0,003774	0	0,498110
pseudonym	0,125430	27,2934	0,003774	0	0,498110
symbol	0,140030	25,6114	0,015094	0	0,492450
woher	0,087514	25,1348	0,007547	0	0,496230

V2 vs. V3	EER	Thr.:	RR	CR	CRR
77993	0,142370	18,8079	0,011321	0	0,494340
pin	0,105540	19,5032	0,011321	0	0,494340
pseudonym	0,077908	25,6772	0,007547	0	0,496230
symbol	0,049057	21,6198	0,015094	0	0,492450
woher	0,075088	25,0508	0,003774	0	0,498110

```
#include "authentication.h"
#include "main.h"

#define DATA_BUF_SIZE 43

// BioHash
#include "biohash.h"

//////////////////////////////////////////////////////////////////////////
// FIFO variables
//////////////////////////////////////////////////////////////////////////

tSampleData sample_fifo[SAMPLE_FIFO_SIZE];
uint8_t fifo_wr = 0;
uint8_t fifo_rd = 0;

//////////////////////////////////////////////////////////////////////////
// USB
//////////////////////////////////////////////////////////////////////////

BYTE InReport[HID_IN_REPORT_LENGTH]; // HID IN  Report => Touch data
BYTE OutReport[HID_OUT_REPORT_LENGTH]; // HID OUT Report => only for tests


//////////////////////////////////////////////////////////////////////////
// Misc. variables
//////////////////////////////////////////////////////////////////////////

BOOL power_up_done = 0;

unsigned char pkt_counter;
unsigned char line_num;
unsigned char last_z;
unsigned char drawing_enabled;
unsigned char check_rotate_button;
```

```
unsigned char restart_touch_read; // This is set from usb_set/get_report
    to initiate reading the ADC
67
68
69 uint8_t dpy_x;
   uint8_t dpy_y;
71
   uint8_t fcount;
73
   // Globale Variablen (nicht mit extern in main.h gekennzeichnet), nur fuer
       die main sichbar
75 struct IntervalMatrix im_ref;
   struct BioHash bioHash_ref;
77 struct BioHash bioHash_cur;
   uint16_t tv[ANZ_MERKMALE];
79
80
81 uint8_t main_cmd = 0;
   uint16_t FrmBaseAddr = 0;
83
84
85
86
87 // Funktion "pausiert" das System um x
   void delay_ms(long time) {
89   unsigned long inner, outer;
90
91   for (outer = 0; outer < time; outer++) {
       // seed the ndrng
93     for (inner = 0; inner < 10000; inner++)
         ;
95   }
96
97 }
```

```
99 ////////////////////////////////////////////////////
   int main(void) {
101   uint8_t i; //Zaehlvariable
      uint8_t j = 0;
103   uint8_t hdtresh = 10;

105   // GUI Komponents //

107   struct DpyButton button2;
      struct DpyButton button3;
109   struct DpyMnuButton m_button1;
      struct DpyMnuButton m_button2;
111   struct DpyMnuButton m_button3;
      struct DpyLine line2;
113   struct DpyLine line3;
      struct DpyLine line4;
115   struct DpyFrame1 mainframe;
      struct DpyFrame1 confframe;
117
      struct DpyRectangle welcRec;
119
      struct DpyButton u_button1;
121   struct DpyButton u_button2;
      struct DpyButton u_button3;
123   struct DpyButton u_button4;
      struct DpyButton u_button5;
125   struct DpyButton u_button6;
      struct DpyButton u_button7;
127   struct DpyButton u_button8;

129   struct DpyButton conf_button1;
      struct DpyButton conf_button2;
131   struct DpyButton conf_button3;
      struct DpyButton conf_button4;
133
```

```
    DpyButton conf_buttons[4];
135 DpyButton buttons[8];
    DpyMnuButton m_buttons[3];
137


139 // Positionierung und Groesseneinteilung der Elemente
    welcRec.posX = 14*CHAR_WIDTH-2;
141 welcRec.posY = 1*CHAR_HEIGHT-1;
    welcRec.width= 20*CHAR_WIDTH+2;
143 welcRec.height=6*CHAR_HEIGHT+1;

145 line2.posX1 = 14*CHAR_WIDTH;
    line2.posY1 = 3*CHAR_HEIGHT;
147 line2.posX2 = 34*CHAR_WIDTH;
    line2.posY2 = 3*CHAR_HEIGHT;
149

    line3.posX1 = 7*CHAR_WIDTH+2; //Verti. Linie fuer das Menue
151 line3.posY1 = 0;
    line3.posX2 = 7*CHAR_WIDTH+2;
153 line3.posY2 = 8*CHAR_HEIGHT;

155 line4.posX1 = 0;//hori. Linie fuer das Menue
    line4.posY1 = 1*CHAR_HEIGHT;
157 line4.posX2 = 7*CHAR_WIDTH+2;
    line4.posY2 = 1*CHAR_HEIGHT;
159

    u_button1.posX = 11*CHAR_WIDTH;
161 u_button1.posY = 2*CHAR_HEIGHT;
    u_button1.name = "User1";
163 u_button1.clicked = FALSE;

165 u_button2.posX = 18*CHAR_WIDTH;
    u_button2.posY = 2*CHAR_HEIGHT;
167 u_button2.name = "User2";
    u_button2.clicked = FALSE;
```

```
169
    u_button3.posX = 25*CHAR_WIDTH;
171 u_button3.posY = 2*CHAR_HEIGHT;
    u_button3.name = "User3";
173 u_button3.clicked = FALSE;

175 u_button4.posX = 32*CHAR_WIDTH;
    u_button4.posY = 2*CHAR_HEIGHT;
177 u_button4.name = "User4";
    u_button4.clicked = FALSE;
179
    u_button5.posX = 11*CHAR_WIDTH;
181 u_button5.posY = 5*CHAR_HEIGHT;
    u_button5.name = "User5";
183 u_button5.clicked = FALSE;

185 u_button6.posX = 18*CHAR_WIDTH;
    u_button6.posY = 5*CHAR_HEIGHT;
187 u_button6.name = "User6";
    u_button6.clicked = FALSE;
189
    u_button7.posX = 25*CHAR_WIDTH;
191 u_button7.posY = 5*CHAR_HEIGHT;
    u_button7.name = "User7";
193 u_button7.clicked = FALSE;

195 u_button8.posX = 32*CHAR_WIDTH;
    u_button8.posY = 5*CHAR_HEIGHT;
197 u_button8.name = "User8";
    u_button8.clicked = FALSE;
199
    button2.posX = 27*CHAR_WIDTH;
201 button2.posY = 7*CHAR_HEIGHT;
    button2.name = "Cancel";
203 button2.clicked = FALSE;
```

```
205     button3.posX = 14*CHAR_WIDTH;
        button3.posY = 7*CHAR_HEIGHT;
207     button3.name = "Ok";
        button3.clicked = FALSE;
209
211     m_button1.posX = 1*CHAR_WIDTH;
        m_button1.posY = 2*CHAR_HEIGHT;
213     m_button1.name = "Verif.";
        m_button1.clicked = FALSE;
215
        m_button2.posX = 1*CHAR_WIDTH;
217     m_button2.posY = 3*CHAR_HEIGHT;
        m_button2.name = "Enrol.";
219     m_button2.clicked = FALSE;

221     m_button3.posX = 1*CHAR_WIDTH;
        m_button3.posY = 4*CHAR_HEIGHT;
223     m_button3.name = "Config";
        m_button3.clicked = FALSE;
225
        conf_button1.posX = 20*CHAR_WIDTH;
227     conf_button1.posY = 1*CHAR_HEIGHT;
        conf_button1.name = "05";
229     conf_button1.clicked = FALSE;

231     conf_button2.posX = 24*CHAR_WIDTH;
        conf_button2.posY = 1*CHAR_HEIGHT;
233     conf_button2.name = "10";
        conf_button2.clicked = TRUE;
235
        conf_button3.posX = 28*CHAR_WIDTH;
237     conf_button3.posY = 1*CHAR_HEIGHT;
        conf_button3.name = "15";
```

```
239   conf_button3.clicked = FALSE;

241   conf_button4.posX = 32*CHAR_WIDTH;
      conf_button4.posY = 1*CHAR_HEIGHT;
243   conf_button4.name = "20";
      conf_button4.clicked = FALSE;
245
      m_buttons[0] = m_button1;
247   m_buttons[1] = m_button2;
      m_buttons[2] = m_button3;
249
      // hinzufuegen der Userbuttons in das array welches spaeter in den frame
          eingebunden wird
251   buttons[0] = u_button1;
      buttons[1] = u_button2;
253   buttons[2] = u_button3;
      buttons[3] = u_button4;
255   buttons[4] = u_button5;
      buttons[5] = u_button6;
257   buttons[6] = u_button7;
      buttons[7] = u_button8;
259
      // hinzufuegen der Configbuttons in das array welches spaeter in den
         frame eingebunden wird
261   conf_buttons[0] = conf_button1;
      conf_buttons[1] = conf_button2;
263   conf_buttons[2] = conf_button3;
      conf_buttons[3] = conf_button4;
265


267   mainframe.buttonLst = buttons;

269   mainframe.buttonLstLen = 8;

271   confframe.buttonLst = conf_buttons;
```

```
273   confframe.buttonLstLen = 4;


275


277   ConfigurePLL();

279   // ETM OFF!
      PINSEL10 = 0;
281

      init_ports();
283   init_serial();
      // Test: disable ADC
285   FIO0SET = SPI_TOUCH_CSN; // P0.16 = SS_TOUCH = 1
      FIO0DIR |= SPI_TOUCH_CSN; // P0.16 = SS_TOUCH
287

      SPI_Init();
289   rtc_init();
      touch_init();
291   init_display();
      init_usb_serial_number();
293   dpy_send_cmd8(DPY_CTRL0 | DPY_CTRL1, DPY_CMD_DISPLAY_REV);

295   // draw version number
      dpy_set_cursor(200, 0);
297   dpy_draw_char('V', DPY_DRAW_INVERTED);
      dpy_draw_num8(USB_DeviceDescriptor[13], DPY_DRAW_INVERTED);
299   dpy_draw_char('.', DPY_DRAW_INVERTED);
      dpy_draw_num8(USB_DeviceDescriptor[12], DPY_DRAW_INVERTED);
301


303   // Delay for Logo Display
      delay_ms(800);
305   dpy_send_cmd8(DPY_CTRL0 | DPY_CTRL1, DPY_CMD_DISPLAY_NORMAL);
      delay_ms(800);
```

```
307
    power_up_done = 1;
309 timer_init();
    touch_calibrate(0);
311
    drawing_enabled = true;
313 LED_ORANGE_OFF
    LED_GREEN_ON
315
#define TOUCH_RX_PLATE  1500.0
317 #define TOUCH_RY_PLATE   150.0

319     dpy_clear();
    dpy_set_cursor(0, 0);
321



323
    set_flash_cur_adr(FLASH_START_ADR_REF); //Schreibadresse der
        Referenzdaten im Flash ROM festlegen
325
    // Anzeige nach der Neuinitialisierung
327 dpy_draw_rect_struct(welcRec);
    dpy_draw_line_struct(line2);
329 dpy_set_cursor(18*CHAR_WIDTH, 2*CHAR_HEIGHT);
    dpy_draw_string(" Welcome!", strlen(" Welcome!"), DPY_DRAW_SOLID);
331 dpy_set_cursor(16*CHAR_WIDTH, 5*CHAR_HEIGHT);
    dpy_draw_string(" NaturaSign...", strlen(" NaturaSign..."),
        DPY_DRAW_SOLID);
333


335 // Dauerschleife
    while(1){
337   dpy_draw_line_struct(line3); //Menue zeichnen
      dpy_draw_line_struct(line4); //Menue zeichnen
339   // Setzt den Cursor
```

```
    dpy_set_cursor(1*CHAR_WIDTH, 0*CHAR_HEIGHT);
341 // Schreibt einen String an Cursorposition
    dpy_draw_string(" MENU", strlen(" MENU"), DPY_DRAW_SOLID);
343 // Setzt den Cursor
    dpy_set_cursor(1*CHAR_WIDTH, 2*CHAR_HEIGHT);
345 dpy_draw_mnuButton_struct(m_buttons[0]);
    dpy_draw_mnuButton_struct(m_buttons[1]);
347 dpy_draw_mnuButton_struct(m_buttons[2]);

349 //Fragt Sensor ab

351 get_next_touch();
    //if(is_touched())
353 //{
      LED_GREEN_ON
355   LED_ORANGE_OFF
      touch_normalize_xy();
357   touch_calc_z();

359   dpy_set_cursor(34*CHAR_WIDTH, 4*CHAR_HEIGHT);

361   //Displaykoordinaten berechnen basierend auf Sensordaten
      dpy_x = touch_x * ((float) DPY_XSIZE / (float) TOUCH_OUTPUT_RANGE_X)
          ;
363   dpy_y = touch_y * ((float) DPY_YSIZE / (float) TOUCH_OUTPUT_RANGE_Y)
          ;


365

      //If Button area is touched, change status
367   //wenn einer der beiden Menupunkte zum Enrollment oder Verifikation
          gewaehlt worden ist
      if(!m_buttons[2].clicked && (m_buttons[0].clicked || m_buttons[1].
          clicked)){
369   // ueberpruefe, dass nich 2 Buttons gleichzeitig aktiv sind
      for(i=0;i<mainframe.buttonLstLen;i++){
```

```
371         if(is_button_touched(mainframe.buttonLst[i], dpy_x, dpy_y)){
              for(j=0; j<mainframe.buttonLstLen; j++){
373           if(buttons[i].name!=buttons[j].name)
              buttons[j].clicked = FALSE;
375         }
            //aktiviere Button falls er vorher inaktiv war und zeichne frame
                neu
377         if(!mainframe.buttonLst[i].clicked){
              mainframe.buttonLst[i].clicked = TRUE;
379           dpy_draw_multiframe_struct(mainframe);
            // deaktiviere Button falls er vorher aktiv war  und zeichne
               frame neu
381         }else
            {
383           mainframe.buttonLst[i].clicked = FALSE;
              dpy_draw_multiframe_struct(mainframe);
385         }
          }
387     }
        }
389
        //If Button area is touched, change status
391     // wenn das Konfigurationsmenu angewaehlt worden ist
        if(m_buttons[2].clicked){
393     for(i=0;i<confframe.buttonLstLen;i++){
          // ueberpruefe, dass nich 2 Buttons gleichzeitig aktiv sind
395       if(is_button_touched(confframe.buttonLst[i], dpy_x, dpy_y)){
              for(j=0; j<mainframe.buttonLstLen; j++){
397           if(conf_buttons[i].name!=conf_buttons[j].name)
              conf_buttons[j].clicked = FALSE;
399         }
            //aktiviere Button falls er vorher inaktiv war, weise Wert zu
               und zeichne Frame neu
401         if(!confframe.buttonLst[i].clicked){
              confframe.buttonLst[i].clicked = TRUE;
```

```
403             // jenachdem welcher Button angeklickt worden ist weise dem
                    Treshold den zugehoerigen Wert zu
                switch(i)
405             {
                    case 0:  hdtresh = 5; break;
                    case 1:  hdtresh = 10; break;
407                 case 2:  hdtresh = 15; break;
                    case 3:  hdtresh = 20; break;
409                 default:  hdtresh = 20; break;
                }
411             dpy_set_cursor(9*CHAR_WIDTH, 1*CHAR_HEIGHT);
                dpy_draw_string("Treshold:", strlen("Treshold:"),
413                 DPY_DRAW_SOLID);
                dpy_draw_multiframe_struct(confframe);
415           //deaktiviere Button falls er vorher inaktiv war und zeichne
                 Frame neu
            }else
            {
417             confframe.buttonLst[i].clicked = FALSE;
                dpy_set_cursor(9*CHAR_WIDTH, 1*CHAR_HEIGHT);
419             dpy_draw_string("Treshold:", strlen("Treshold:"),
                    DPY_DRAW_SOLID);
421             dpy_draw_multiframe_struct(confframe);
            }
423       }
        }
425     }

427     //If m_Button area is touched, change status
        for(i=0; i<3; i++){
429       if(is_mnuButton_touched(m_buttons[i], dpy_x, dpy_y)){
            if(!m_buttons[i].clicked){
431           m_buttons[i].clicked = TRUE;
            }else
            {
433
```

```
            m_buttons[i].clicked = FALSE;
435         }
            for(j=0; j<3; j++){
437         if(m_buttons[i].name!=m_buttons[j].name)
            m_buttons[j].clicked = FALSE;
439         }
            // zeige Userframe falls Enr. oder Ver. Menu ausgewaehlt wurde
441         if(!m_buttons[2].clicked && (m_buttons[0].clicked || m_buttons
              [1].clicked)){
            dpy_clear();
443           dpy_draw_multiframe_struct(mainframe);
            }
445         //ansonsten zeige das Konfigurationsmenu an
            else{
447           dpy_clear();
            dpy_set_cursor(9*CHAR_WIDTH, 1*CHAR_HEIGHT);
449         dpy_draw_string("Treshold:", strlen("Treshold:"),
              DPY_DRAW_SOLID);
            dpy_draw_multiframe_struct(confframe);
451         }
          }
453     }

455     // sollte keiner der der Menupunkte gewaehlt worden sein zeige den
          Willkommensbildschirm an
        if(!m_buttons[0].clicked && !m_buttons[1].clicked && !m_buttons[2].
          clicked){
457       dpy_clear();
          dpy_draw_rect_struct(welcRec);
459       dpy_draw_line_struct(line2);
          dpy_set_cursor(18*CHAR_WIDTH, 2*CHAR_HEIGHT);
461       dpy_draw_string(" Welcome!", strlen(" Welcome!"), DPY_DRAW_SOLID);
          dpy_set_cursor(16*CHAR_WIDTH, 5*CHAR_HEIGHT);
463       dpy_draw_string(" NaturaSign...", strlen(" NaturaSign..."),
            DPY_DRAW_SOLID);
```

```
      }
465
467
// ------------ ENROLLMENT ---------------------------//
469 // sollte der Nutzer sich in Menupunkt Enr. befinden und der Button fuer
    User1 geklickt worden sein
     if(buttons[0].clicked && m_buttons[1].clicked){ // Enrollment
471     dpy_clear();
        delay_ms(180); // Fuer 180 ms den gedrueckten Button darstellen
473     buttons[0].clicked = FALSE;
        m_buttons[1].clicked  = FALSE;
475     // rufe funktion aus authentication.c auf
        enrollment(&im_ref, &bioHash_ref, FLASH_START_ADR_REF);
477  }

479 // ------------------- Verifikation ---------------------//
// sollte der Nutzer sich in Menupunkt Ver. befinden und der Button fuer
    User1 geklickt worden sein
481  if(buttons[0].clicked && m_buttons[0].clicked ){ // Verifikation
        dpy_clear();
483     delay_ms(180); // Fuer 180 ms den gedrueckten Button darstellen
        buttons[0].clicked = FALSE;
485     m_buttons[0].clicked  = FALSE;
        // rufe funktion aus authentication.c auf
487     verification(&im_ref, &bioHash_ref, &bioHash_cur,
           FLASH_START_ADR_REF, hdtresh);
     }
489 }

491 }
```

src/main.cpp

4.2 authentication.c

```
verification(&im_ref, &bioHash_ref, &bioHash_cur, FLASH_START_ADR_REF,
    hdtresh){
...
  struct DpyLine line2;
  struct DpyRectangle verRec;

  verRec.posX = 14*CHAR_WIDTH-2;
  verRec.posY = 1*CHAR_HEIGHT-1;
  verRec.width= 20*CHAR_WIDTH+2;
  verRec.height=6*CHAR_HEIGHT+1;

  line2.posX1 = 14*CHAR_WIDTH;
  line2.posY1 = 3*CHAR_HEIGHT;
  line2.posX2 = 34*CHAR_WIDTH;
  line2.posY2 = 3*CHAR_HEIGHT;
...
 if(hd<hdtresh){
  dpy_draw_rect_struct(verRec);
  dpy_draw_line_struct(line2);
  dpy_set_cursor(17*CHAR_WIDTH, 2*CHAR_HEIGHT);
  dpy_draw_string(" Verification!", strlen(" Verification!"),
      DPY_DRAW_SOLID);
  dpy_set_cursor(18*CHAR_WIDTH, 5*CHAR_HEIGHT);
  dpy_draw_string(" successful!", strlen(" successful!"), DPY_DRAW_SOLID);
 }
 else{
  dpy_draw_rect_struct(verRec);
  dpy_draw_line_struct(line2);
  dpy_set_cursor(17*CHAR_WIDTH, 2*CHAR_HEIGHT);
  dpy_draw_string(" Verification!", strlen(" Verification!"),
      DPY_DRAW_SOLID);
  dpy_set_cursor(20*CHAR_WIDTH, 5*CHAR_HEIGHT);
  dpy_draw_string(" failed!", strlen(" failed!"), DPY_DRAW_SOLID);
 }
```

```
    ...
33  }
```

src/veri.cpp