

```

1  /*
2
3  Test program for sending measured values from the
4  Raspberry PI over Modbus TCP to the web server
5
6  Sending is done with HTTP POST
7
8  Josef Bernhardt 13.2.2021
9
10
11 Hardware Raspberry PI 4
12
13 Codeblocks fuer Raspbian
14
15 Example Message
16
17 POST /heizung/sqlabfrage.php HTTP/1.0
18 Host: www.bernhardt.de
19 Connection: close
20 Content-Type: application/x-www-form-urlencoded;charset=UTF-8;
21 Content-Length: 44
22
23 aktion=write&file=inputs.csv&inhalt=23.00;1;0;1;0;1;0;1;1
24
25 */
26 #include <stdio.h>          /* printf, sprintf */
27 #include <stdlib.h>         /* exit */
28 #include <unistd.h>         /* read, write, close */
29 #include <string.h>         /* memcpy, memset */
30 #include <sys/socket.h>     /* socket, connect */
31 #include <netinet/in.h>     /* struct sockaddr_in, struct sockaddr */
32 #include <netdb.h>          /* struct hostent, gethostbyname */
33
34 #define FC1 1
35 #define FC2 2
36
37
38 void error(const char *msg) { perror(msg); /*exit(0);*/ }
39
40 // String for Result from Modbus Call FC01
41 char strresult[16] = {'0',';','0',';','0',';','0',';','0',';','0',';','0',';','0',';','0','0'};
42
43 // Post Message to HTTP Server
44 char *msg_post = "POST /modbus/sqlabfrage.php HTTP/1.0\r\n";
45 char *msg_host = "Host: www.bernhardt.de\r\n";
46 char *msg_conn = "Connection: close\r\n";
47 char *msg_cont = "Content-Type: application/x-www-form-urlencoded;charset=UTF-8;\r\n";
48 char *msg_conl = "Content-Length: %d\r\n\r\n";
49
50 char *msg_acti_fc1 = "aktion=write&file=outputs.csv&inhalt=%.2f;";
51 char *msg_acti_fc2 = "aktion=write&file=inputs.csv&inhalt=%.2f;";
52 // mit Content length
53 char msg_conl_r[128]={0};
54 // mit Zahlenwert
55 char msg_acti_r[128]={0};
56
57 char msg_send[512]={0};
58
59 //char message[1024];
60 char response[1024];
61
62 // Data from Modbus
63 char usr_response[128]={0};
64
65 // build Poststring with Modbus Data and float Value
66 void build_post_string_fc1(float value, char *str)
67 {
68     // Clear Post message memory
69     memset(&msg_send,0,512);
70     // Insert float Value
71     sprintf(msg_acti_r,msg_acti_fc1,value);
72
73     // Calculate and insert the length of the action string

```

```

74     sprintf(msg_conl_r,"Content-Length: %d\r\n\r\n",strlen(msg_acti_fc1)+15);
75     // Build Post String
76     strcat(msg_send,msg_post);
77     strcat(msg_send,msg_host);
78     strcat(msg_send,msg_conn);
79     strcat(msg_send,msg_cont);
80     strcat(msg_send,msg_conl_r);
81     strcat(msg_send,msg_acti_r);
82     strcat(msg_send,str);
83 }
84
85 // build Poststring with Modbus Data and float Value
86 void build_post_string_fc2(float value, char *str)
87 {
88     // Clear Post message memory
89     memset(&msg_send,0,512);
90     // Insert float Value
91     sprintf(msg_acti_r,msg_acti_fc2,value);
92
93     // Calculate and insert the length of the action string
94     sprintf(msg_conl_r,"Content-Length: %d\r\n\r\n",strlen(msg_acti_fc2)+15);
95     // Build Post String
96     strcat(msg_send,msg_post);
97     strcat(msg_send,msg_host);
98     strcat(msg_send,msg_conn);
99     strcat(msg_send,msg_cont);
100    strcat(msg_send,msg_conl_r);
101    strcat(msg_send,msg_acti_r);
102    strcat(msg_send,str);
103 }
104
105
106 // Send Post Message to Server
107 void sendpostmsg()
108 {
109
110     int portno =         80;
111     char *host =         "www.bernhardt.de";
112
113     struct hostent *server;
114     struct sockaddr_in serv_addr;
115     int sockfd, bytes, sent, received, total;
116
117     /* create the socket */
118     sockfd = socket(AF_INET, SOCK_STREAM, 0);
119     if (sockfd < 0) error("ERROR opening socket");
120
121     /* lookup the ip address */
122     server = gethostbyname(host);
123     if (server == NULL) error("ERROR, no such host");
124
125     /* fill in the structure */
126     memset(&serv_addr,0,sizeof(serv_addr));
127     serv_addr.sin_family = AF_INET;
128     serv_addr.sin_port = htons(portno);
129     memcpy(&serv_addr.sin_addr.s_addr,server->h_addr,server->h_length);
130
131     /* connect the socket */
132     if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
133         error("ERROR connecting");
134     /* send the request */
135     total = strlen(msg_send);
136     sent = 0;
137
138     bytes = write(sockfd,msg_send+sent,total-sent);
139     /*
140     do {
141         bytes = write(sockfd,msg_send+sent,total-sent);
142         if (bytes < 0)
143             error("ERROR writing message to socket");
144         if (bytes == 0)
145             break;
146         sent+=bytes;

```

```

147         } while (sent < total);
148     */
149     sleep(0.020);
150
151     /* receive the response */
152     memset(response,0,sizeof(response));
153     total = sizeof(response)-1;
154     received = 0;
155
156     // Read Answer from HTTP Webserver
157     bytes = read(sockfd,response,total);
158     /*
159     do {
160         bytes = read(sockfd,response+received,total-received);
161         if (bytes < 0)
162             error("ERROR reading response from socket");
163         if (bytes == 0)
164             break;
165         received+=bytes;
166     } while (received < total);
167
168     if (received == total)
169         error("ERROR storing complete response from socket");
170     */
171
172     /* close the socket */
173     close(sockfd);
174 }
175
176
177 // Read Data from raspberry PI PLC
178 void sendtcpip_modbusFC1()
179 {
180
181     unsigned char obuf[128];
182     unsigned char ibuf[128];
183
184     unsigned short reg_no = 0;
185     unsigned short num_regs = 8;
186     unsigned short unit = 1;
187     unsigned short transid = 1;
188     unsigned short slaveadr=1;
189
190     int portno = 502;
191
192     // test with second Raspberry PI
193     // char *host = "192.168.178.89";
194     // local address
195     char *host = "127.0.0.1";
196
197     struct hostent *server;
198     struct sockaddr_in serv_addr;
199     int sockfd, bytes, sent, received, total,i;
200
201     unsigned char mask = 0x01;
202     unsigned char result = 0x09;
203
204     obuf[0] = (unsigned char)transid >>8;
205     obuf[1] = (unsigned char)transid;
206     obuf[2] = 0;
207     obuf[3] = 0;
208     obuf[4] = 0;
209     obuf[5] = 6; // Message Length Low(6 bytes to follow)
210     obuf[6] = slaveadr; // SlaveAddress
211     obuf[7] = FC1; // Function
212     obuf[8] = reg_no >> 8; // Starting Address High
213     obuf[9] = reg_no & 0xff; // Starting Address Low
214     obuf[10] = num_regs >> 8; // Quantity of Coils High
215     obuf[11] = num_regs & 0xff; // Quantity of Coils Low
216
217     /* create the socket */
218     sockfd = socket(AF_INET, SOCK_STREAM, 0);
219     if (sockfd < 0) error("ERROR opening socket");

```

```

220
221     /* lookup the ip address */
222     server = gethostbyname(host);
223     if (server == NULL) error("ERROR, no such host");
224
225     /* fill in the structure */
226     memset(&serv_addr,0,sizeof(serv_addr));
227     serv_addr.sin_family = AF_INET;
228     serv_addr.sin_port = htons(portno);
229     memcpy(&serv_addr.sin_addr.s_addr,server->h_addr,server->h_length);
230
231     /* connect the socket */
232     if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
233         error("ERROR connecting");
234     /* send the request */
235     total = 12;
236
237     bytes = write(sockfd,obuf,total);
238     if (bytes < 0) error("ERROR writing message to socket");
239
240     /*
241     printf("TCP-Send..\r\n");
242     for (i=0;i<bytes;i++)
243     {
244         printf(" %02x ",obuf[i]);
245     }
246     printf("\r\n");
247 */
248     sleep (0.05) ;
249
250     bytes = read(sockfd,ibuf,128);
251     //printf("Bytes: %d\r\n",bytes);
252     //printf("TCP-Received..\r\n");
253
254     for (i=0;i<bytes;i++)
255     {
256         //printf(" %02x ",ibuf[i]);
257     }
258     /*
259     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
260     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
261     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
262     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
263     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
264     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
265     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
266     if ( ibuf[result] & mask ) printf("1;"); else printf("0;");
267 */
268
269     if ( ibuf[result] & mask ) strresult[0] = '1'; else strresult[0] = '0'; mask
= mask << 1;
270     if ( ibuf[result] & mask ) strresult[2] = '1'; else strresult[2] = '0'; mask
= mask << 1;
271     if ( ibuf[result] & mask ) strresult[4] = '1'; else strresult[4] = '0'; mask
= mask << 1;
272     if ( ibuf[result] & mask ) strresult[6] = '1'; else strresult[6] = '0'; mask
= mask << 1;
273     if ( ibuf[result] & mask ) strresult[8] = '1'; else strresult[8] = '0'; mask
= mask << 1;
274     if ( ibuf[result] & mask ) strresult[10] = '1'; else strresult[10] = '0';
mask = mask << 1;
275     if ( ibuf[result] & mask ) strresult[12] = '1'; else strresult[12] = '0';
mask = mask << 1;
276     if ( ibuf[result] & mask ) strresult[14] = '1'; else strresult[14] = '0';
277
278     //printf("%s\r\n",strresult);
279     /* close the socket */
280     close(sockfd);
281 }
282
283
284
285 // Read Data from raspberry PI PLC Input Pins %IX0.0 to %IX0.7

```

```

286 void sendtcpip_modbusFC2()
287 {
288
289     unsigned char obuf[128];
290     unsigned char ibuf[128];
291
292     unsigned short reg_no = 0;
293     unsigned short num_regs = 8;
294     unsigned short unit = 1;
295     unsigned short transid = 1;
296     unsigned short slaveadr = 1;
297
298     int portno = 502;
299
300     // test with second Raspberry PI
301     // char *host = "192.168.178.89";
302     // local address
303     char *host = "127.0.0.1";
304
305     struct hostent *server;
306     struct sockaddr_in serv_addr;
307     int sockfd, bytes, sent, received, total, i;
308
309     unsigned char mask = 0x01;
310     unsigned char result = 0x09;
311
312     obuf[0] = (unsigned char)transid >> 8;
313     obuf[1] = (unsigned char)transid;
314     obuf[2] = 0;
315     obuf[3] = 0;
316     obuf[4] = 0;
317     obuf[5] = 6; // Message Length Low(6 bytes to follow)
318     obuf[6] = slaveadr; // SlaveAddress
319     obuf[7] = FC2; // Function
320     obuf[8] = reg_no >> 8; // Starting Address High
321     obuf[9] = reg_no & 0xff; // Starting Address Low
322     obuf[10] = num_regs >> 8; // Quantity of Coils High
323     obuf[11] = num_regs & 0xff; // Quantity of Coils Low
324
325     /* create the socket */
326     sockfd = socket(AF_INET, SOCK_STREAM, 0);
327     if (sockfd < 0) error("ERROR opening socket");
328
329     /* lookup the ip address */
330     server = gethostbyname(host);
331     if (server == NULL) error("ERROR, no such host");
332
333     /* fill in the structure */
334     memset(&serv_addr, 0, sizeof(serv_addr));
335     serv_addr.sin_family = AF_INET;
336     serv_addr.sin_port = htons(portno);
337     memcpy(&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
338
339     /* connect the socket */
340     if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
341         error("ERROR connecting");
342     /* send the request */
343     total = 12;
344
345     bytes = write(sockfd, obuf, total);
346     if (bytes < 0) error("ERROR writing message to socket");
347
348     /*
349     printf("TCP-Send..\r\n");
350     for (i=0; i<bytes; i++)
351     {
352         printf(" %02x ", obuf[i]);
353     }
354     printf("\r\n");
355 */
356     sleep (0.05) ;
357
358     bytes = read(sockfd, ibuf, 128);

```

```

359         //printf("Bytes: %d\r\n",bytes);
360         //printf("TCP-Received..\r\n");
361
362         for (i=0;i<bytes;i++)
363         {
364             //printf(" %02x ",ibuf[i]);
365         }
366     /*
367         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
368         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
369         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
370         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
371         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
372         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
373         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");mask = mask << 1;
374         if ( ibuf[result] & mask ) printf("1;"); else printf("0;");
375     */
376
377         if ( ibuf[result] & mask ) strresult[0] = '1'; else strresult[0] = '0'; mask
= mask << 1;
378         if ( ibuf[result] & mask ) strresult[2] = '1'; else strresult[2] = '0'; mask
= mask << 1;
379         if ( ibuf[result] & mask ) strresult[4] = '1'; else strresult[4] = '0'; mask
= mask << 1;
380         if ( ibuf[result] & mask ) strresult[6] = '1'; else strresult[6] = '0'; mask
= mask << 1;
381         if ( ibuf[result] & mask ) strresult[8] = '1'; else strresult[8] = '0'; mask
= mask << 1;
382         if ( ibuf[result] & mask ) strresult[10] = '1'; else strresult[10] = '0';
mask = mask << 1;
383         if ( ibuf[result] & mask ) strresult[12] = '1'; else strresult[12] = '0';
mask = mask << 1;
384         if ( ibuf[result] & mask ) strresult[14] = '1'; else strresult[14] = '0';
385
386
387         //printf("%s\r\n",strresult);
388         /* close the socket */
389         close(sockfd);
390     }
391
392     // function to replace , with . or others
393     int replacechar(char *str, char orig, char rep) {
394         char *ix = str;
395         int n = 0;
396         while((ix = strchr(ix, orig)) != NULL) {
397             *ix++ = rep;
398             n++;
399         }
400         return n;
401     }
402
403
404     // main function
405     int main(int argc,char *argv[])
406     {
407
408         printf("Testsoftware to Read Modbus Data %QX0.0 to QX0.7\r\n");
409         printf("from Raspberry PI openplcproject PLC\r\n");
410         printf("\r\n");
411         printf("Send Data to HTTP Server www.bernhardt.de \r\n");
412         printf("to php Server directory /modbus/sqlabfrage.php \r\n");
413         printf("\r\n");
414
415         float x1 = 0.0;
416         float x2 = 5.0;
417
418         while(1)
419         {
420             // Teststring when no Modbus Data available
421             // build_post_string("1.34;1;0;1;0;1;0;1;0");
422
423             x1=x1+0.1;
424             x2=x2+0.1;

```

```

425
426     if (x1 > 4.9) x1 = 0;
427     if (x2 > 9.9) x2 = 5;
428
429     // Read Data from Modbus
430     sendtcpip_modbusFC1();
431     // Generate Poststring
432     build_post_string_fc1(x1,strresult);
433     // Send Data to Webserver PHP Software
434     sendpostmsg();
435
436     // printf(msg_send);printf("\r\n");
437
438     printf(strresult);printf(" ");
439     // Wait
440     sleep(0.5);
441
442     // Read Data from Modbus
443     sendtcpip_modbusFC2();
444     // Generate Poststring
445     build_post_string_fc2(x2,strresult);
446     // Send Data to Webserver PHP Software
447     sendpostmsg();
448     // printf(msg_send);
449
450     sleep(0.5);
451     // Answer from Webserver
452     // printf(response);
453
454     // Reasponse from Modbus
455     printf(strresult);
456
457     // Send Post Message to Terminal Windows
458     // printf(msg_send);
459     printf("\r\n");
460
461
462
463 }
464 return 0;
465 }
466

```