```c
/*
 * Embedded.c
 *
 *  Created on: 31 Jan 2018
 *      Author: cp5g15 & jte1n17
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <avr/power.h>
#include <inttypes.h>
#include <math.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "lcd.h"
#include "pictor.h"
#include "fonts/Mash.h"

#define FREQ 488
#define TONE_PRESCALER 1

char charge[]="Charging";
char off[]="Off      ";
char discharge[]="Discharge";
char A_per_S[] =" As ";
char blank[]="  ";
float *battery_value;
int bit=10;

// Interfacing Function
////////////////////////////////////////////////////////////////////////////////////////////////////
// Drawing Tool 1 - Drawing Rectangle Pixel============================================
void draw_rect(int leftside,int rightside, int topside, int bottomside,uint16_t col)
{
    rectangle r;
        r.left=leftside;
        r.right=rightside;
        r.top=topside;
        r.bottom=bottomside;
        fill_rectangle(r,col);
}

// Drawing Tool 2 - Drawing Square Pixel=============================================
void draw_square(int x, int y,int w,uint16_t colour)
{
    draw_rect(x,x+w,y,y+w,colour);
}

// Drawing 1 - Vertical Line=========================================================
void draw_vline(int x, int y,int z)
{
    rectangle r;
        r.left=x;
        r.right=y;
        r.top=z;
        r.bottom=z;
```

```
58          fill_rectangle(r,WHITE);
59  }
60
61  // Drawing 2 - Horizontal
      Line=================================================================
62  void draw_hline(int x, int y,int z)
63  {
64      rectangle r;
65          r.left=z;
66          r.right=z;
67          r.top=x;
68          r.bottom=y;
69          fill_rectangle(r,WHITE);
70  }
71
72  // Drawing 3 - Smart Meter
      Box==================================================================
73  void draw_smeter(void){
74      draw_rect(10,45,25,60,SKYBLUE);
75      draw_rect(10,45,25,26,WHITE);
76      draw_rect(10,45,59,60,WHITE);
77      draw_rect(10,11,25,60,WHITE);
78      draw_rect(44,45,25,60,WHITE);
79      //side
80      draw_rect(45,46,24,59,SKYBLUE);
81      draw_rect(46,47,23,58,SKYBLUE);
82      draw_rect(47,48,22,57,SKYBLUE);
83      draw_rect(48,49,21,56,SKYBLUE);
84      draw_rect(49,50,20,55,SKYBLUE);
85      draw_rect(50,51,19,54,SKYBLUE);
86      draw_rect(51,52,18,53,SKYBLUE);
87      draw_rect(52,53,17,52,SKYBLUE);
88      draw_rect(53,54,16,51,SKYBLUE);
89      draw_rect(54,55,15,50,SKYBLUE);
90      draw_rect(55,56,14,49,WHITE);
91      //whiteside
92      draw_rect(45,46,58,59,WHITE);
93      draw_rect(46,47,57,58,WHITE);
94      draw_rect(47,48,56,57,WHITE);
95      draw_rect(48,49,55,56,WHITE);
96      draw_rect(49,50,54,55,WHITE);
97      draw_rect(50,51,53,54,WHITE);
98      draw_rect(51,52,52,53,WHITE);
99      draw_rect(52,53,51,52,WHITE);
100     draw_rect(53,54,50,51,WHITE);
101     draw_rect(54,55,49,50,WHITE);
102     //top
103     draw_rect(11,46,24,25,WHITE);
104     draw_rect(12,47,23,24,WHITE);
105     draw_rect(13,48,22,23,WHITE);
106     draw_rect(14,49,21,22,WHITE);
107     draw_rect(15,50,20,21,WHITE);
108     draw_rect(16,51,19,20,WHITE);
109     draw_rect(17,52,18,19,WHITE);
110     draw_rect(18,53,17,18,WHITE);
111     draw_rect(19,54,16,17,WHITE);
112     draw_rect(20,55,15,16,WHITE);
113     draw_rect(21,56,14,15,WHITE);
114     //white
115     draw_rect(12,45,24,25,SKYBLUE);
116     draw_rect(13,46,23,24,SKYBLUE);
117     draw_rect(14,47,22,23,SKYBLUE);
```

```
118     draw_rect(15,48,21,22,SKYBLUE);
119     draw_rect(16,49,20,21,SKYBLUE);
120     draw_rect(17,50,19,20,SKYBLUE);
121     draw_rect(18,51,18,19,SKYBLUE);
122     draw_rect(19,52,17,18,SKYBLUE);
123     draw_rect(20,53,16,17,SKYBLUE);
124     draw_rect(21,54,15,16,SKYBLUE);
125     //hook
126     draw_rect(35,42,9,14,WHITE);
127     draw_rect(36,41,6,9,WHITE);
128     draw_rect(37,40,4,5,WHITE);
129     //frame
130     draw_rect(27,38,35,48,CHOCO);
131     draw_rect(28,36,37,46,BLACK);
132 }
133
134 // Drawing 4 -
    Battery================================================================================
135 void draw_batteryframe(void){
136     draw_rect(10,35,268,308,WHITE); //frame
137     draw_rect(18,27,263,268,WHITE); //top polar
138     draw_rect(18,27,308,313,WHITE); //bottom polar
139 }
140
141 void draw_batterypolar(uint16_t colour){
142     draw_rect(12,33,270,306,colour); //colour frame
143     draw_rect(17,28,298,301,WHITE); //minus sign
144     draw_rect(17,28,278,281,WHITE); //plus sign vertical
145     draw_rect(21,24,274,285,WHITE); //plus sign horizontal
146 }
147
148 // Drawing 5 - Busbar Design
    (Lightning)============================================================
149 void draw_busbar_dark(){ //Remove all drawing on the busbar column
150     draw_rect(9,29,105,136,BLACK);
151 }
152
153 void draw_busbar(uint16_t colour){ //Full Lightning (Animation 1)
154     //top half
155     draw_rect(25,26,105,106,colour);
156     draw_rect(24,26,106,107,colour);
157     draw_rect(23,25,107,108,colour);
158     draw_rect(22,25,108,109,colour);
159     draw_rect(21,24,109,110,colour);
160     draw_rect(20,24,110,111,colour);
161     draw_rect(19,23,111,112,colour);
162     draw_rect(18,23,112,113,colour);
163     draw_rect(17,22,113,114,colour);
164     draw_rect(16,22,114,115,colour);
165     draw_rect(15,21,115,116,colour);
166     draw_rect(14,21,116,117,colour);
167     draw_rect(13,20,117,118,colour);
168     //middle half
169     draw_rect(12,29,118,119,colour);
170     draw_rect(11,28,119,120,colour);
171     draw_rect(10,27,120,121,colour);
172     draw_rect(9,26,121,122,colour);
173     //last half
174     draw_rect(17,25,122,123,colour);
175     draw_rect(17,24,123,124,colour);
176     draw_rect(16,23,124,125,colour);
177     draw_rect(16,22,125,126,colour);
```

```
178     draw_rect(15,21,126,127,colour);
179     draw_rect(15,20,127,128,colour);
180     draw_rect(14,19,128,129,colour);
181     draw_rect(14,18,129,130,colour);
182     draw_rect(13,17,130,131,colour);
183     draw_rect(13,16,131,132,colour);
184     draw_rect(12,15,132,133,colour);
185     draw_rect(12,14,133,134,colour);
186     draw_rect(11,13,134,135,colour);
187     draw_rect(11,12,135,136,colour);
188     }
189
190 void draw_busbar2(uint16_t colour){ //Bottom slice of Lightning (Animation 2)
191     draw_rect(14,19,105,106,colour);
192     draw_rect(14,18,106,107,colour);
193     draw_rect(13,17,107,108,colour);
194     draw_rect(13,16,108,109,colour);
195     draw_rect(12,15,109,110,colour);
196     draw_rect(12,14,110,111,colour);
197     draw_rect(11,13,111,112,colour);
198     draw_rect(11,12,112,113,colour);
199     }
200
201 void draw_busbar3(uint16_t colour){ //Half of Lightning (Animation 3)
202     //middle half
203     draw_rect(12,29,105,106,colour);
204     draw_rect(11,28,106,107,colour);
205     draw_rect(10,27,107,108,colour);
206     draw_rect(9,26,108,109,colour);
207     //last half
208     draw_rect(17,25,109,110,colour);
209     draw_rect(17,24,110,111,colour);
210     draw_rect(16,23,111,112,colour);
211     draw_rect(16,22,112,113,colour);
212     draw_rect(15,21,113,114,colour);
213     draw_rect(15,20,114,115,colour);
214     draw_rect(14,19,115,116,colour);
215     draw_rect(14,18,116,117,colour);
216     draw_rect(13,17,117,118,colour);
217     draw_rect(13,16,118,119,colour);
218     draw_rect(12,15,119,120,colour);
219     draw_rect(12,14,120,121,colour);
220     draw_rect(11,13,121,122,colour);
221     draw_rect(11,12,122,123,colour);
222 }
223
224 void draw_busbar4(uint16_t colour){ //Three Quarter of Lightning (Animation 4)
225     draw_rect(19,23,105,106,colour);
226     draw_rect(18,23,106,107,colour);
227     draw_rect(17,22,107,108,colour);
228     draw_rect(16,22,108,109,colour);
229     draw_rect(15,21,109,110,colour);
230     draw_rect(14,21,110,111,colour);
231     draw_rect(13,20,111,112,colour);
232     //middle half
233     draw_rect(12,29,112,113,colour);
234     draw_rect(11,28,113,114,colour);
235     draw_rect(10,27,114,115,colour);
236     draw_rect(9,26,115,116,colour);
237     //last half
238     draw_rect(17,25,116,117,colour);
239     draw_rect(17,24,117,118,colour);
```

```
240     draw_rect(16,23,118,119,colour);
241     draw_rect(16,22,119,120,colour);
242     draw_rect(15,21,120,121,colour);
243     draw_rect(15,20,121,122,colour);
244     draw_rect(14,19,122,123,colour);
245     draw_rect(14,18,123,124,colour);
246     draw_rect(13,17,124,125,colour);
247     draw_rect(13,16,125,126,colour);
248     draw_rect(12,15,126,127,colour);
249     draw_rect(12,14,127,128,colour);
250     draw_rect(11,13,128,129,colour);
251     draw_rect(11,12,129,130,colour);
252 }
253
254 //Drawing 6 - Mains Design (A
    socket)=================================================================
255 void draw_mains(void){
256     draw_rect(125,145,107,129,WHITE);
257     draw_rect(128,132,122,125,BLACK);
258     draw_rect(138,142,122,125,BLACK);
259     draw_rect(134,136,115,120,BLACK);
260     draw_rect(140,143,110,115,BLACK);
261     draw_rect(141,142,111,114,RED);
262 }
263
264 //Drawing 7 - Wind Design (3 slices of
    wind)=======================================================
265 void draw_wind(uint16_t colour){
266     //Top slice of the wind=============================================
267     draw_rect(34,40,166,167,colour);
268     draw_rect(30,44,167,168,colour);
269     draw_rect(27,47,168,169,colour);
270     draw_rect(24,49,169,170,colour);
271     draw_rect(22,82,170,171,colour);
272     draw_rect(20,36,171,172,colour);
273     draw_rect(18,32,172,173,colour);
274     draw_rect(17,28,173,174,colour);
275     draw_rect(16,25,174,175,colour);
276     draw_rect(15,22,175,176,colour);
277     draw_rect(14,20,176,177,colour);
278     draw_rect(14,18,177,178,colour);
279     draw_rect(13,16,178,179,colour);
280     draw_rect(13,15,179,180,colour);
281     draw_rect(13,14,180,181,colour);
282     draw_rect(55,80,171,172,colour);
283     draw_rect(57,77,172,173,colour);
284     draw_rect(60,74,173,174,colour);
285     draw_rect(64,70,174,175,colour);
286     draw_rect(68,84,179,170,colour);
287     draw_rect(72,86,168,169,colour);
288     draw_rect(76,87,167,168,colour);
289     draw_rect(79,88,166,167,colour);
290     draw_rect(82,89,165,166,colour);
291     draw_rect(84,90,164,165,colour);
292     draw_rect(86,90,163,164,colour);
293     draw_rect(88,91,162,163,colour);
294     draw_rect(89,91,161,162,colour);
295     draw_rect(90,91,160,161,colour);
296     //Middle slice of the wind=========================================
297     draw_rect(34,40,176,177,colour);
298     draw_rect(30,44,177,178,colour);
299     draw_rect(27,47,178,179,colour);
```

```
300     draw_rect(24,49,179,180,colour);
301     draw_rect(22,82,180,181,colour);
302     draw_rect(20,36,181,182,colour);
303     draw_rect(18,32,182,183,colour);
304     draw_rect(17,28,183,184,colour);
305     draw_rect(16,25,184,185,colour);
306     draw_rect(15,22,185,186,colour);
307     draw_rect(14,20,186,187,colour);
308     draw_rect(14,18,187,188,colour);
309     draw_rect(13,16,188,189,colour);
310     draw_rect(13,15,189,190,colour);
311     draw_rect(13,14,190,191,colour);
312     draw_rect(55,80,181,182,colour);
313     draw_rect(57,77,182,183,colour);
314     draw_rect(60,74,183,184,colour);
315     draw_rect(64,70,184,185,colour);
316     draw_rect(68,84,189,180,colour);
317     draw_rect(72,86,178,179,colour);
318     draw_rect(76,87,177,178,colour);
319     draw_rect(79,88,176,177,colour);
320     draw_rect(82,89,175,176,colour);
321     draw_rect(84,90,174,175,colour);
322     draw_rect(86,90,173,174,colour);
323     draw_rect(88,91,172,173,colour);
324     draw_rect(89,91,171,172,colour);
325     draw_rect(90,91,170,171,colour);
326     //Bottom slice of the wind=========================================
327     draw_rect(34,40,186,187,colour);
328     draw_rect(30,44,187,188,colour);
329     draw_rect(27,47,188,189,colour);
330     draw_rect(24,49,189,190,colour);
331     draw_rect(22,82,190,191,colour);
332     draw_rect(20,36,191,192,colour);
333     draw_rect(18,32,192,193,colour);
334     draw_rect(17,28,193,194,colour);
335     draw_rect(16,25,194,195,colour);
336     draw_rect(15,22,195,196,colour);
337     draw_rect(14,20,196,197,colour);
338     draw_rect(14,18,197,198,colour);
339     draw_rect(13,16,198,199,colour);
340     draw_rect(13,15,199,200,colour);
341     draw_rect(13,14,200,201,colour);
342     draw_rect(55,80,191,192,colour);
343     draw_rect(57,77,192,193,colour);
344     draw_rect(60,74,193,194,colour);
345     draw_rect(64,70,194,195,colour);
346     draw_rect(68,84,189,190,colour);
347     draw_rect(72,86,188,189,colour);
348     draw_rect(76,87,187,188,colour);
349     draw_rect(79,88,186,187,colour);
350     draw_rect(82,89,185,186,colour);
351     draw_rect(84,90,184,185,colour);
352     draw_rect(86,90,183,184,colour);
353     draw_rect(88,91,182,183,colour);
354     draw_rect(89,91,181,182,colour);
355     draw_rect(90,91,180,181,colour);
356 }
357
358 void draw_wind2(uint16_t colour){
359     //Top slice of the wind=========================================
360     draw_rect(30,33,167,168,colour);
361     draw_rect(27,33,168,169,colour);
```

```
362     draw_rect(24,33,169,170,colour);
363     draw_rect(22,33,170,171,colour);
364     draw_rect(20,33,171,172,colour);
365     draw_rect(18,32,172,173,colour);
366     draw_rect(17,28,173,174,colour);
367     draw_rect(16,25,174,175,colour);
368     draw_rect(15,22,175,176,colour);
369     draw_rect(14,20,176,177,colour);
370     draw_rect(14,18,177,178,colour);
371     draw_rect(13,16,178,179,colour);
372     draw_rect(13,15,179,180,colour);
373     draw_rect(13,14,180,181,colour);
374     //Middle slice of the wind=========================================
375     draw_rect(30,33,177,178,colour);
376     draw_rect(27,33,178,179,colour);
377     draw_rect(24,33,179,180,colour);
378     draw_rect(22,33,180,181,colour);
379     draw_rect(20,33,181,182,colour);
380     draw_rect(18,32,182,183,colour);
381     draw_rect(17,28,183,184,colour);
382     draw_rect(16,25,184,185,colour);
383     draw_rect(15,22,185,186,colour);
384     draw_rect(14,20,186,187,colour);
385     draw_rect(14,18,187,188,colour);
386     draw_rect(13,16,188,189,colour);
387     draw_rect(13,15,189,190,colour);
388     draw_rect(13,14,190,191,colour);
389     //Bottom slice of the wind=========================================
390     draw_rect(30,33,187,188,colour);
391     draw_rect(27,33,188,189,colour);
392     draw_rect(24,33,189,190,colour);
393     draw_rect(22,33,190,191,colour);
394     draw_rect(20,33,191,192,colour);
395     draw_rect(18,32,192,193,colour);
396     draw_rect(17,28,193,194,colour);
397     draw_rect(16,25,194,195,colour);
398     draw_rect(15,22,195,196,colour);
399     draw_rect(14,20,196,197,colour);
400     draw_rect(14,18,197,198,colour);
401     draw_rect(13,16,198,199,colour);
402     draw_rect(13,15,199,200,colour);
403     draw_rect(13,14,200,201,colour);
404 }
405
406 void draw_wind3(uint16_t colour){
407     //Top slice of the wind============================================
408     draw_rect(34,40,166,167,colour);
409     draw_rect(30,44,167,168,colour);
410     draw_rect(27,47,168,169,colour);
411     draw_rect(24,49,169,170,colour);
412     draw_rect(22,53,170,171,colour);
413     draw_rect(20,36,171,172,colour);
414     draw_rect(18,32,172,173,colour);
415     draw_rect(17,28,173,174,colour);
416     draw_rect(16,25,174,175,colour);
417     draw_rect(15,22,175,176,colour);
418     draw_rect(14,20,176,177,colour);
419     draw_rect(14,18,177,178,colour);
420     draw_rect(13,16,178,179,colour);
421     draw_rect(13,15,179,180,colour);
422     draw_rect(13,14,180,181,colour);
423     //Middle slice of the wind=========================================
```

```
424     draw_rect(34,40,176,177,colour);
425     draw_rect(30,44,177,178,colour);
426     draw_rect(27,47,178,179,colour);
427     draw_rect(24,49,179,180,colour);
428     draw_rect(22,53,180,181,colour);
429     draw_rect(20,36,181,182,colour);
430     draw_rect(18,32,182,183,colour);
431     draw_rect(17,28,183,184,colour);
432     draw_rect(16,25,184,185,colour);
433     draw_rect(15,22,185,186,colour);
434     draw_rect(14,20,186,187,colour);
435     draw_rect(14,18,187,188,colour);
436     draw_rect(13,16,188,189,colour);
437     draw_rect(13,15,189,190,colour);
438     draw_rect(13,14,190,191,colour);
439     //Bottom slice of the wind=========================================
440     draw_rect(34,40,186,187,colour);
441     draw_rect(30,44,187,188,colour);
442     draw_rect(27,47,188,189,colour);
443     draw_rect(24,49,189,190,colour);
444     draw_rect(22,53,190,191,colour);
445     draw_rect(20,36,191,192,colour);
446     draw_rect(18,32,192,193,colour);
447     draw_rect(17,28,193,194,colour);
448     draw_rect(16,25,194,195,colour);
449     draw_rect(15,22,195,196,colour);
450     draw_rect(14,20,196,197,colour);
451     draw_rect(14,18,197,198,colour);
452     draw_rect(13,16,198,199,colour);
453     draw_rect(13,15,199,200,colour);
454     draw_rect(13,14,200,201,colour);
455 }
456
457 void draw_wind4(uint16_t colour){
458     //Top slice of the wind============================================
459     draw_rect(34,40,166,167,colour);
460     draw_rect(30,44,167,168,colour);
461     draw_rect(27,47,168,169,colour);
462     draw_rect(24,49,169,170,colour);
463     draw_rect(22,73,170,171,colour);
464     draw_rect(20,36,171,172,colour);
465     draw_rect(18,32,172,173,colour);
466     draw_rect(17,28,173,174,colour);
467     draw_rect(16,25,174,175,colour);
468     draw_rect(15,22,175,176,colour);
469     draw_rect(14,20,176,177,colour);
470     draw_rect(14,18,177,178,colour);
471     draw_rect(13,16,178,179,colour);
472     draw_rect(13,15,179,180,colour);
473     draw_rect(13,14,180,181,colour);
474     draw_rect(55,73,171,172,colour);
475     draw_rect(57,73,172,173,colour);
476     draw_rect(60,73,173,174,colour);
477     draw_rect(64,70,174,175,colour);
478     draw_rect(68,73,179,170,colour);
479     draw_rect(72,73,168,169,colour);
480     //Middle slice of the wind=========================================
481     draw_rect(34,40,176,177,colour);
482     draw_rect(30,44,177,178,colour);
483     draw_rect(27,47,178,179,colour);
484     draw_rect(24,49,179,180,colour);
485     draw_rect(22,73,180,181,colour);
```

```
    draw_rect(20,36,181,182,colour);
    draw_rect(18,32,182,183,colour);
    draw_rect(17,28,183,184,colour);
    draw_rect(16,25,184,185,colour);
    draw_rect(15,22,185,186,colour);
    draw_rect(14,20,186,187,colour);
    draw_rect(14,18,187,188,colour);
    draw_rect(13,16,188,189,colour);
    draw_rect(13,15,189,190,colour);
    draw_rect(13,14,190,191,colour);
    draw_rect(55,73,181,182,colour);
    draw_rect(57,73,182,183,colour);
    draw_rect(60,73,183,184,colour);
    draw_rect(64,70,184,185,colour);
    draw_rect(68,73,189,180,colour);
    draw_rect(72,73,178,179,colour);
    //Bottom slice of the wind=========================================
    draw_rect(34,40,186,187,colour);
    draw_rect(30,44,187,188,colour);
    draw_rect(27,47,188,189,colour);
    draw_rect(24,49,189,190,colour);
    draw_rect(22,73,190,191,colour);
    draw_rect(20,36,191,192,colour);
    draw_rect(18,32,192,193,colour);
    draw_rect(17,28,193,194,colour);
    draw_rect(16,25,194,195,colour);
    draw_rect(15,22,195,196,colour);
    draw_rect(14,20,196,197,colour);
    draw_rect(14,18,197,198,colour);
    draw_rect(13,16,198,199,colour);
    draw_rect(13,15,199,200,colour);
    draw_rect(13,14,200,201,colour);
    draw_rect(55,73,191,192,colour);
    draw_rect(57,73,192,193,colour);
    draw_rect(60,73,193,194,colour);
    draw_rect(64,70,194,195,colour);
    draw_rect(68,73,189,190,colour);
    draw_rect(72,73,188,189,colour);
}

//Drawing 8 - Solar Design (Circle with 8
   triangles)=============================================
void draw_solar(uint16_t colour){
    //1st triangle=================================================
    draw_rect(183,184,169,170,colour);
    draw_rect(182,185,170,171,colour);
    draw_rect(181,186,171,172,colour);
    draw_rect(180,187,172,173,colour);
    //2nd triangle=================================================
    draw_rect(183,184,196,197,colour);
    draw_rect(182,185,195,196,colour);
    draw_rect(181,186,194,195,colour);
    draw_rect(180,187,193,194,colour);
    //3rd triangle=================================================
    draw_rect(172,173,180,187,colour);
    draw_rect(171,172,181,186,colour);
    draw_rect(170,171,182,185,colour);
    draw_rect(169,170,183,184,colour);
    //4th triangle=================================================
    draw_rect(194,195,180,187,colour);
    draw_rect(195,196,181,186,colour);
    draw_rect(196,197,182,185,colour);
```

```
547     draw_rect(197,198,183,184,colour);
548     //5th triangle=============================================================
549     draw_rect(190,195,171,172,colour);
550     draw_rect(191,195,172,173,colour);
551     draw_rect(192,195,173,174,colour);
552     draw_rect(193,195,174,175,colour);
553     draw_rect(194,195,175,176,colour);
554     //6th triangle=============================================================
555     draw_rect(190,195,195,196,colour);
556     draw_rect(191,195,194,195,colour);
557     draw_rect(192,195,193,194,colour);
558     draw_rect(193,195,192,193,colour);
559     draw_rect(194,195,191,192,colour);
560     //7th triangle=============================================================
561     draw_rect(170,171,191,192,colour);
562     draw_rect(170,172,192,193,colour);
563     draw_rect(170,173,193,194,colour);
564     draw_rect(170,174,194,195,colour);
565     draw_rect(170,175,195,196,colour);
566     //8th triangle=============================================================
567     draw_rect(170,175,171,172,colour);
568     draw_rect(170,174,172,173,colour);
569     draw_rect(170,173,173,174,colour);
570     draw_rect(170,172,174,175,colour);
571     draw_rect(170,171,175,176,colour);
572     //circle===================================================================
573     draw_rect(181,186,177,179,colour);
574     draw_rect(179,188,179,181,colour);
575     draw_rect(177,190,181,185,colour);
576     draw_rect(179,188,185,187,colour);
577     draw_rect(181,186,187,189,colour);
578     }
579
580 void draw_solar2(uint16_t colour){ //slanted triangle(for animation)====
581     //circle===================================================================
582     draw_rect(181,186,177,179,colour);
583     draw_rect(179,188,179,181,colour);
584     draw_rect(177,190,181,185,colour);
585     draw_rect(179,188,185,187,colour);
586     draw_rect(181,186,187,189,colour);
587     //1st triangle=============================================================
588     draw_rect(178,179,169,170,colour);
589     draw_rect(177,180,170,171,colour);
590     draw_rect(177,181,171,172,colour);
591     draw_rect(177,182,172,173,colour);
592     draw_rect(176,179,173,174,colour);
593     draw_rect(176,177,174,175,colour);
594     //2nd triangle=============================================================
595     draw_rect(188,189,169,170,colour);
596     draw_rect(187,190,170,171,colour);
597     draw_rect(186,190,171,172,colour);
598     draw_rect(185,190,172,173,colour);
599     draw_rect(188,191,173,174,colour);
600     draw_rect(190,191,174,175,colour);
601     //3rd triangle=============================================================
602     draw_rect(177,178,196,197,colour);
603     draw_rect(176,179,195,196,colour);
604     draw_rect(176,180,194,195,colour);
605     draw_rect(176,181,193,194,colour);
606     draw_rect(175,178,192,193,colour);
607     draw_rect(175,176,191,192,colour);
608     //4th triangle=============================================================
```

```
609     draw_rect(188,189,196,197,colour);
610     draw_rect(187,190,195,196,colour);
611     draw_rect(186,190,194,195,colour);
612     draw_rect(185,190,193,194,colour);
613     draw_rect(188,191,192,193,colour);
614     draw_rect(190,191,191,192,colour);
615     //5th triangle=============================================================
616     draw_rect(172,173,176,179,colour);
617     draw_rect(171,172,176,181,colour);
618     draw_rect(170,171,177,181,colour);
619     draw_rect(168,170,177,180,colour);
620     draw_rect(167,168,178,179,colour);
621     //6th triangle=============================================================
622     draw_rect(172,173,186,189,colour);
623     draw_rect(171,172,184,189,colour);
624     draw_rect(170,171,184,188,colour);
625     draw_rect(168,170,185,188,colour);
626     draw_rect(167,168,186,187,colour);
627     //7th triangle=============================================================
628     draw_rect(193,194,176,179,colour);
629     draw_rect(194,195,176,181,colour);
630     draw_rect(195,196,177,181,colour);
631     draw_rect(196,198,177,180,colour);
632     draw_rect(198,199,178,179,colour);
633     //8th triangle=============================================================
634     draw_rect(193,194,186,189,colour);
635     draw_rect(194,195,184,189,colour);
636     draw_rect(195,196,184,188,colour);
637     draw_rect(196,198,185,188,colour);
638     draw_rect(198,199,186,187,colour);
639     }
640
641 // Drawing 9 - Smart Meter word
    (Animation)=============================================================
642 void draw_smart(uint16_t colour1,uint16_t colour2,uint16_t colour3,uint16_t
    colour4,uint16_t colour5){
643     pictorDrawS("S", (point){70,10}, colour1, BLACK , Mash,3);
644     pictorDrawS("M", (point){93,10}, colour2, BLACK , Mash,3);
645     pictorDrawS("A", (point){116,10}, colour3, BLACK , Mash,3);
646     pictorDrawS("R", (point){139,10}, colour4, BLACK , Mash,3);
647     pictorDrawS("T", (point){161,10}, colour5, BLACK , Mash,3);
648 }
649
650 void draw_meter(uint16_t colour1,uint16_t colour2,uint16_t colour3,uint16_t
    colour4,uint16_t colour5){
651     pictorDrawS("M", (point){70,40}, colour1, BLACK , Mash,3);
652     pictorDrawS("E", (point){93,40}, colour2, BLACK , Mash,3);
653     pictorDrawS("T", (point){116,40}, colour3, BLACK , Mash,3);
654     pictorDrawS("E", (point){139,40}, colour4, BLACK , Mash,3);
655     pictorDrawS("R", (point){161,40}, colour5, BLACK , Mash,3);
656     pictorDrawS("G", (point){190,15}, WHITE, BLACK , Mash,6);
657 }
658
659 //Function for
    battery////////////////////////////////////////////////////////////////////////
660 //Charging state=============================================================
661 void charging(void)
662 {
663    *battery_value = *battery_value + 1;
664     draw_batterypolar(YELLOW);
665     PORTD |= _BV(1);// On Charge battery
666     PORTD &= ~_BV(3);// Off discharge battery
```

```
667     display_value(44,273,charge,blank); //Battery charging
668 }
669
670 //Discharging state=====================================================
671 void discharging(void)
672 {
673     *battery_value = *battery_value - 1;
674     draw_batterypolar(GREEN);
675     PORTD &= ~_BV(1);// Off Charge battery
676     PORTD |= _BV(3);// On discharge battery
677     display_value(44,273,discharge,blank); //Battery discharging
678 }
679
680 //Battery turn off=====================================================
681 void battery_off(void)
682 {
683     draw_batterypolar(BLACK);
684     PORTD &= ~_BV(1);// Off Charge battery
685     PORTD &= ~_BV(3);// Off discharge battery
686     display_value(44,273,off,blank); //Battery Off
687 }
688
689 //Display battery value=================================================
690 void battery_display(void)
691 {
692     char battery_value_string[bit];
693     int actual_battery=*battery_value;
694     itoa(actual_battery,battery_value_string,10);
695     display_value(58,293,battery_value_string,A_per_S);//Battery Value
696 }
697
698 // IL MATTO SYSTEM FUNCTION
    ////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////
699
700 void init_adc(void){
701     ADMUX = 0;
702     ADMUX |= _BV(ADLAR); /* F_ADC = F_CPU/64 */
703     ADCSRA |= _BV(ADEN); /* Enable ADC */
704     ADCSRA |= _BV(ADPS2) | _BV(ADPS1);
705 }
706
707 void channel_adc(uint8_t n)
708 {
709     ADMUX= n;
710 }
711
712 uint16_t read_adc(void)
713 {
714     ADCSRA |= _BV(ADSC); /* Start Conversions */
715     while (ADCSRA&_BV(ADSC));
716     return ADC;
717 }
718
719 void display_description(int x_axis, int y_axis, char description[]){
720     display.x=x_axis;
721     display.y=y_axis;
722     display_string(description);
723 }
724
725 void display_value(int x_axis, int y_axis, char value[], char unit[]){
726     display.x=x_axis;
```

```
727     display.y=y_axis;
728     display_string(value);
729     display_string(unit);
730 }
731 
732 void init_pwm(void)
733 {
734     TCCR2A =_BV(COM2A1)|_BV(WGM20);//
735     TCCR2B= _BV(CS20);
736     OCR2A=0;
737     DDRD |= _BV(7);
738 }
739 
740 //Main Function
741 int main ()
742 {
743     // PIN & IL MATTO INITIALISATION
    ////////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////
744     //Power COmsumption Management
745     power_spi_disable();
746 
747     //Initialise adc
748     init_adc();
749 
750     //LCD Initialise
751     init_pwm();
752     init_lcd();
753 
754     // Pin Configuration
755 
756     DDRA &= ~_BV(1);
757     DDRA &= ~_BV(3);
758     DDRA &= ~_BV(5);
759     DDRD |= _BV(0);
760     DDRD |= _BV(2);
761     DDRD |= _BV(4);
762 
763     DDRD |= _BV(1);
764     DDRD |= _BV(3);
765 
766 
767     // Value Initialisation
    ////////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////
768     //voltage
769     int *voltage;
770     uint16_t resultvoltage;
771     uint32_t busvoltage,busvoltagedec;
772     voltage=(int*)malloc(bit*sizeof(int));
773 
774     //current
775     int *current;
776     uint16_t resultcurrent;
777     uint32_t buscurrent,buscurrentdec;
778     current=(int*)malloc(bit*sizeof(int));
779 
780        //power and
    energy*************************************************************************************
    ***************************
781     int *power;
782     uint16_t power_new,power_old=0;
```

```
783     power=(float*)malloc(bit*sizeof(float));
784 
785     //wind
786     uint16_t resultwind;
787     float *wind_value;
788     float wind_value_new,wind_value_old=0;
789     wind_value=(float*)malloc(bit*sizeof(float));
790 
791     //solar
792     float *solar_value;
793     uint16_t resultsolar;
794     float solar_value_new,solar_value_old=0;
795     solar_value=(float*)malloc(bit*sizeof(float));
796 
797     //main
798     float *main_value;
799     float main_value_new,main_value_old=0;
800     main_value=(float*)malloc(bit*sizeof(float));
801 
802     //Battery
803     battery_value=(float*)malloc(bit*sizeof(float));
804 
805     // LAYOUT STRING STORAGE
    ////////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////
806     //Title
807     char title[]="TEAM G SMART METER";
808 
809     //Description
810     char Busbar[]="=BUSBAR=";
811     char VBusbar[]="V:";
812     char IBusbar[]="I:";
813     char Wind[]="=WIND=";
814     char Solar[]="=SOLAR=";
815     char Mains[]="=MAINS=";
816     char Using[]="Using:";
817     char Used[]="Used:";
818     char Battery[]="=BATTERY=";
819     char Load1[]="Load 1:";
820     char Load2[]="Load 2:";
821     char Load3[]="Load 3:";
822     char Total[]="Total:";
823     char currentlabel[]="I:";
824     char Load[]="=LOAD=";
825     char currentt[]="Current:";
826     char power1[]="Power:";
827     char energy1[]="Energy:";
828 
829     //Value representation String
830     char high[]="On ";
831     char low[] ="Off";
832 
833     //Unit String
834     char Vrms[] = " V  ";
835     char Irms[] =" mA  ";
836     char Ampere[] =" A   ";
837     char cWh[] =" kWh   ";
838     char Watt[] = " W  ";
839 
840     //INTERFACE
    LAYOUT////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////////////
```

```
841     // Title Team G Smart Meter
842     draw_smeter();
843     draw_vline(1,239,85);
844     draw_vline(1,239,145);
845     draw_vline(1,239,238);
846     draw_hline(85,145,120);
847     draw_hline(145,320,120);
848 
849     //Busbar Representation Section
850     display_description(37,90,Busbar);
851     display_description(42,110,VBusbar);
852     display_description(42,125,IBusbar);
853 
854     //Mains Representation Section
855     draw_mains();
856     display_description(165,90,Mains);
857     display_description(151,110,Using);
858     display_description(151,125,Used);
859 
860     //wind Representation Section
861     display_description(42,150,Wind); //Wind
862     display_description(10,210,currentt);
863     display_description(10,225,Total);//Total wind
864 
865     //solar Representation Section
866     display_description (165,150,Solar); // Solar
867     display_description(135,210,currentt);
868     display_description(135,225,Total);//Total solar
869 
870     //Battery Representation Section
871     display_description(38,245,Battery);
872     draw_batteryframe();
873     display_description(44,293,currentlabel);
874 
875     //Load Representation Section
876     display_description(171,245,Load);
877     draw_square(223,263,10,WHITE);
878     draw_square(223,283,10,WHITE);
879     draw_square(223,303,10,WHITE);
880     display_description(140,265,Load1); //Load1
881     display_description(140,285,Load2);//Load2
882     display_description(140,305,Load3); //Load3
883 
884     display_description(10,75,power1);
885     display_description(130,75,energy1);
886 
887     // Value String Storage
   ////////////////////////////////////////////////////////////////////////////////
   ////////////////////////////////////////////////////////////
888     //Value+Unit display
889     char voltagestring[bit];
890     char voltagestringdecimal[bit];
891     char currentstring[bit];
892     char currentstringdecimal[bit];
893     char windstring[bit];
894     char windstringdecimal[bit];
895     char totalwindstring[bit];
896     char totalwindstringdecimal[bit];
897     char solarstring[bit];
898     char solarstringdecimal[bit];
899     char totalsolarstring[bit];
900     char totalsolarstringdecimal[bit];
```

```
901     char mainstring[bit];
902     char mainstringdecimal[bit];
903     char totalmainstring[bit];
904     char totalmainstringdecimal[bit];
905     char powerstring[bit];
906     char powerstringdecimal[bit];
907     char totalpowerstring[bit];
908     char totalpowerstringdecimal[bit];
909     char onedecimalpoint[]=".";
910     char twodecimalpoint[]=".0";
911     char threedecimalpoint[]=".00";
912 
913     //initialize battery
914     int tot_battery_value = 0;
915     *battery_value=0;
916     battery_off();
917     int i=0;
918     int j=87;
919 
920     // CONVERTION SYSTEM
    ////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////////
921     for(;;){
922         //convert to string
923         //voltage bus
924         channel_adc(0);
925         for(i=0;i<=9;i++){
926             float Max_voltage=0;
927             resultvoltage=read_adc();
928             if(resultvoltage>0){
929             *voltage=(resultvoltage*3.3/1024)*1.55*1000;
930             if (*voltage>=Max_voltage) {
931 
932                 Max_voltage=*current;
933                }
934             else
935               {*voltage=Max_voltage;}
936 
937             }
938             else{*voltage=0;}
939             _delay_ms(1);
940         }
941 
942         busvoltage=*voltage/1000;
943         busvoltagedec=*voltage%1000;
944         itoa(busvoltage,voltagestring,10);
945         itoa(busvoltagedec,voltagestringdecimal,10);
946         if(busvoltagedec<10)
947         {
948             strcat(voltagestring,threedecimalpoint);
949             strcat(voltagestring,voltagestringdecimal);
950         }
951         else if(busvoltagedec<100 &&busvoltagedec>=10)
952         {
953             strcat(voltagestring,twodecimalpoint);
954             strcat(voltagestring,voltagestringdecimal);
955         }
956         else
957         {
958             strcat(voltagestring,onedecimalpoint);
959             strcat(voltagestring,voltagestringdecimal);
960         }
```

```
961
962         display_value(55,110,voltagestring,Vrms); //VBus Bar
963
964         ADMUX =0;
965         channel_adc(2);
966         resultcurrent=0;
967
968         for(i=0;i<=9;i++){
969             float Max_current=0;
970             resultcurrent=read_adc();
971             if(resultcurrent>0){
972             *current=(((resultcurrent*3.3/1024))*2.6627+0.7092)*1000;
973
974                 if (*current>=Max_current) {
975
976                     Max_current=*current;
977                 }
978                 else
979                 {*current=Max_current;}
980             }
981
982         else{*current=0;}
983
984             _delay_ms(1);
985         }
986         buscurrent=*current/1000;
987         buscurrentdec=*current%1000;
988         itoa(buscurrent,currentstring,10);
989         itoa(buscurrentdec,currentstringdecimal,10);
990         if(buscurrentdec<10)
991         {
992             strcat(currentstring,threedecimalpoint);
993             strcat(currentstring,currentstringdecimal);
994         }
995         else if(buscurrentdec<100 &&buscurrentdec>=10)
996         {
997             strcat(currentstring,twodecimalpoint);
998             strcat(currentstring,currentstringdecimal);
999         }
1000        else
1001        {
1002            strcat(currentstring,onedecimalpoint);
1003            strcat(currentstring,currentstringdecimal);
1004        }
1005        display_value(55,125,currentstring,Ampere); //IBusBar
1006
1007        ADMUX =0;
1008
1009        *power=(*current)*(*voltage);
1010        int buspower=*power/1000;
1011        int buspowerdec=*power%1000;
1012        itoa(buspower,powerstring,10);
1013        itoa(buspowerdec,powerstringdecimal,10);
1014        if(buspowerdec<10)
1015        {
1016            strcat(powerstring,threedecimalpoint);
1017            strcat(powerstring,powerstringdecimal);
1018        }
1019        else if(buspowerdec<100 &&buspowerdec>=10)
1020        {
1021            strcat(powerstring,twodecimalpoint);
1022            strcat(powerstring,powerstringdecimal);
```

```
1023        }
1024        else
1025        {
1026            strcat(powerstring,onedecimalpoint);
1027            strcat(powerstring,powerstringdecimal);
1028        }
1029        display_value(60,75,powerstring,Watt); //IBusBar
1030
1031        //wind
1032        resultwind=0;
1033        channel_adc(4);
1034        resultwind=read_adc();
1035        *wind_value=(resultwind*3.3/1024)*2.05*0.511;
1036        int windpow3=(*wind_value)*1000;
1037        int actual_wind=windpow3/1000;
1038        int winddec = windpow3%1000;
1039        itoa(actual_wind,windstring,10);
1040        itoa(winddec,windstringdecimal,10);
1041        if(winddec<10)
1042        {
1043            strcat(windstring,threedecimalpoint);
1044            strcat(windstring,windstringdecimal);
1045        }
1046        else if(winddec<100 &&winddec>=10)
1047        {
1048            strcat(windstring,twodecimalpoint);
1049            strcat(windstring,windstringdecimal);
1050        }
1051        else
1052        {
1053            strcat(windstring,onedecimalpoint);
1054            strcat(windstring,windstringdecimal);
1055        }
1056        display_value(60,210,windstring,Ampere); //Wind
1057
1058        ADMUX = 0;
1059
1060        //solar
1061        *solar_value=0;
1062        channel_adc(6);
1063        resultsolar=read_adc();
1064        *solar_value=(resultsolar*3.3/1024)*1.1;
1065        int solarpow3=(*solar_value)*1000;
1066        uint32_t actual_solar=solarpow3/1000;
1067        uint32_t solardec = solarpow3%1000;
1068        itoa(actual_solar,solarstring,10);
1069        itoa(solardec,solarstringdecimal,10);
1070        if(solardec<10)
1071        {
1072            strcat(solarstring,threedecimalpoint);
1073            strcat(solarstring,solarstringdecimal);
1074        }
1075        else if(solardec<100 &&solardec>=10)
1076        {
1077            strcat(solarstring,twodecimalpoint);
1078            strcat(solarstring,solarstringdecimal);
1079        }
1080        else
1081        {
1082            strcat(solarstring,onedecimalpoint);
1083            strcat(solarstring,solarstringdecimal);
1084        }
```

```
1085        display_value(185,210,solarstring,Ampere); // Solar
1086
1087        //load1 theater
1088        float actual_load1;
1089        if(PINA & _BV(1)){
1090            draw_square(225,265,6,GREEN);
1091            actual_load1=0.8;
1092            PORTD |= _BV(0);
1093            display_value(190,265,high,blank); //Battery On/Off
1094        }
1095        else{
1096            draw_square(225,265,6,RED);
1097            actual_load1=0;
1098            PORTD &= ~_BV(0);
1099            display_value(190,265,low,blank); //Battery On/Off
1100        }
1101
1102        //load2 life support
1103        float actual_load2;
1104        if(PINA & _BV(3)){
1105            draw_square(225,285,6,GREEN);
1106            actual_load2=1.8;
1107            PORTD |= _BV(2);
1108            display_value(190,285,high,blank); //Battery On/Off
1109        }
1110        else{
1111            draw_square(225,285,6,RED);
1112            actual_load2=0;
1113            PORTD &= ~_BV(2);
1114            display_value(190,285,low,blank); //Battery On/Off
1115        }
1116
1117        //load3 ward
1118        float actual_load3;
1119        if(PINA & _BV(5)){
1120            draw_square(225,305,6,GREEN);
1121            PORTD |= _BV(4);
1122            actual_load3=1.4;
1123            display_value(190,305,high,blank); //Battery On/Off
1124        }
1125        else{
1126            draw_square(225,305,6,RED);
1127            actual_load3=0;
1128            PORTD &= ~_BV(4);
1129            display_value(190,305,low,blank); //Battery On/Off
1130        }
1131
1132        float totalinput= *wind_value+*solar_value;
1133        float totalload = actual_load1+actual_load2+actual_load3;
1134        float left= totalinput-totalload;
1135        *main_value=0;
1136
1137
1138        //calculation=========================================================================
    ================================
1139        if(busvoltage<=0.05){
1140            *main_value=0;
1141            battery_off();
1142        }
1143        else{
1144            if(j<87){
1145            j++;
```

```
1146            }
1147            else{
1148                power_new=(*power/60)+power_old;
1149                power_old=power_new;
1150                uint32_t totalpowerk=power_new*10;
1151                uint32_t totalpower= totalpowerk/10;
1152                uint32_t totalpowerdec= totalpowerk%10; //ISsue in insert decimal point
1153                itoa(totalpower,totalpowerstring,10);
1154                itoa(totalpowerdec,totalpowerstringdecimal,10);
1155                strcat(totalpowerstring,onedecimalpoint);
1156                strcat(totalpowerstring,totalpowerstringdecimal);
1157
1158                display_value(180,75,totalpowerstring,cWh);//energy display with battery
1159
1160               if (left>=1){
1161              *main_value=0;
1162              charging();
1163                }
1164              else if (left>0 && left<1){
1165              *main_value=1-left;
1166              charging();
1167              }
1168              else if (left==0){
1169                 if  (*battery_value>10){
1170                  *main_value=0;
1171                  battery_off();
1172                  }
1173                  else{
1174                  *main_value=1;
1175                   charging();
1176                   }
1177              }
1178              else if (left<0 && left>=(-1)){// -1<=left<0
1179                  if (*battery_value>10 && left<=(-0.9)){
1180                  *main_value=0;
1181                  discharging();
1182                   }
1183                  else{
1184                    if(*battery_value>0){
1185                    *main_value=-left;
1186                    battery_off();
1187                     }
1188                   else{
1189                     *main_value=1-left;
1190                      charging();
1191                    }
1192              }
1193            }
1194              else if(left<=(-1) && left>=(-2)){//-2<=left<-1
1195                   if(*battery_value>10){
1196                   *main_value = -left-1;
1197                    discharging();
1198                   }
1199                   else if(*battery_value>=10){
1200                   *main_value = -left;
1201                   battery_off();
1202                   }
1203                   else{
1204                   *main_value = -left+1;
1205                   charging();
1206                    }
1207              }
```

```
1208              else if(left<=(-2) && left>=(-3)){//-3<=left<-2
1209                 if(*battery_value>10){
1210                *main_value = -left-1;
1211                discharging();
1212                 }
1213                else{
1214                *main_value = -left;
1215                battery_off();
1216                 }
1217            }
1218              else if(left<=(-3) && left>=(-4)){//-4<=left<-3
1219                 if(*battery_value>=1){
1220                discharging();
1221                *main_value = -left - 1;
1222                 }
1223                else{
1224                     battery_off();
1225                   if(actual_load1==0){//when load 1 is off
1226                    *main_value = 1.8;
1227                    }
1228                   else{
1229                    *main_value = 2.6;
1230                    }
1231                   PORTD &= ~_BV(4);//turn off load 3
1232                  }
1233                }
1234              battery_display();
1235              OCR2A=((*main_value*255)/3.3)/1.9;
1236              j=0;
1237
1238            int mainpow3=(*main_value)*1000;
1239            uint32_t actual_main=mainpow3/1000;
1240            uint32_t maindec = mainpow3%1000;
1241            itoa(actual_main,mainstring,10);
1242            itoa(maindec,mainstringdecimal,10);
1243            if(maindec<10)
1244                {
1245                strcat(mainstring,threedecimalpoint);
1246                strcat(mainstring,mainstringdecimal);
1247                }
1248            else if(maindec<100 &&maindec>=10)
1249                {
1250                strcat(mainstring,twodecimalpoint);
1251                strcat(mainstring,mainstringdecimal);
1252                }
1253            else
1254                {
1255                strcat(mainstring,onedecimalpoint);
1256                strcat(mainstring,mainstringdecimal);
1257                }
1258                display_value(188,110,mainstring,Ampere); //Main
1259              }
1260            //main used
1261            main_value_new=(*main_value)+main_value_old;
1262            main_value_old = main_value_new;
1263            uint32_t totalmaink=main_value_new*10;
1264            uint32_t totalmain= (totalmaink)/10;
1265            uint32_t totalmaindec=(totalmaink)%10;
1266            itoa(totalmain,totalmainstring,10);
1267            itoa(totalmaindec,totalmainstringdecimal,10);
1268            strcat(totalmainstring,onedecimalpoint);
1269            strcat(totalmainstring,totalmainstringdecimal);
```

```
1270            display_value(188,125,totalmainstring,A_per_S); //Mains Used
1271
1272            //total solar
1273        solar_value_new=(*solar_value)+solar_value_old;
1274        solar_value_old = solar_value_new;
1275        uint32_t totalsolark=solar_value_new*10;
1276        uint32_t totalsolar= (totalsolark)/10;
1277        uint32_t totalsolardec=0; //Issue in insert decimal point
1278        itoa(totalsolar,totalsolarstring,10);
1279        itoa(totalsolardec,totalsolarstringdecimal,10);
1280        strcat(totalsolarstring,onedecimalpoint);
1281        strcat(totalsolarstring,totalsolarstringdecimal);
1282
1283        display_value(185,225,totalsolarstring,A_per_S);//Total solar
1284
1285        //total wind
1286        wind_value_new=(*wind_value)+wind_value_old;
1287        wind_value_old = wind_value_new;
1288        uint32_t totalwindk=wind_value_new*10;
1289        uint32_t totalwind= totalwindk/10;
1290        uint32_t totalwinddec= totalwindk%10; //ISsue in insert decimal point
1291        itoa(totalwind,totalwindstring,10);
1292        itoa(totalwinddec,totalwindstringdecimal,10);
1293        strcat(totalwindstring,onedecimalpoint);
1294        strcat(totalwindstring,totalwindstringdecimal);
1295
1296        display_value(60,225,totalwindstring,A_per_S);//Total wind
1297        }
1298
1299        //Animation with 0.5 second delay
    ============================================================
1300        draw_wind(BLACK);
1301        draw_wind2(BLUE);
1302        draw_smart(WHITE,CYAN,GREEN,RED,MAGENTA);
1303        draw_meter(MAGENTA,WHITE,CYAN,GREEN,RED);
1304        draw_busbar_dark();
1305        draw_busbar2(YELLOW);
1306        draw_solar2(BLACK);
1307        draw_solar(RED);
1308        _delay_ms(100);
1309        draw_wind3(BLUE);
1310        draw_smart(MAGENTA,WHITE,CYAN,GREEN,RED);
1311        draw_meter(RED,MAGENTA,WHITE,CYAN,GREEN);
1312        draw_busbar_dark();
1313        draw_busbar3(YELLOW);
1314        draw_solar(BLACK);
1315        draw_solar2(ORANGE);
1316        _delay_ms(100);
1317        draw_wind4(BLUE);
1318        draw_smart(RED,MAGENTA,WHITE,CYAN,GREEN);
1319        draw_meter(GREEN,RED,MAGENTA,WHITE,CYAN);
1320        draw_busbar_dark();
1321        draw_busbar4(YELLOW);
1322        draw_solar2(BLACK);
1323        draw_solar(YELLOW);
1324        _delay_ms(100);
1325        draw_wind(BLUE);
1326        draw_smart(GREEN,RED,MAGENTA,WHITE,CYAN);
1327        draw_meter(CYAN,GREEN,RED,MAGENTA,WHITE);
1328        draw_busbar_dark();
1329        draw_busbar(YELLOW);
1330        draw_solar(BLACK);
```

```
1331        draw_solar2(RED);
1332        _delay_ms(100);
1333        draw_smart(CYAN,GREEN,RED,MAGENTA,WHITE);
1334        draw_meter(WHITE,CYAN,GREEN,RED,MAGENTA);
1335        draw_solar2(BLACK);
1336        draw_solar(ORANGE);
1337        _delay_ms(100);
1338        draw_smart(WHITE,WHITE,WHITE,WHITE,WHITE);
1339        draw_meter(WHITE,WHITE,WHITE,WHITE,WHITE);
1340        draw_solar(BLACK);
1341        draw_solar2(YELLOW);
1342    }
1343
1344    //Free Memory
1345    free(voltage);
1346    free(current);
1347    free(battery_value);
1348    free(wind_value);
1349    free(solar_value);
1350    free(main_value);
1351    free(power);
1352
1353    while(1);
1354    return 0;
1355 }
1356
```