```
1
2  #include <Arduino.h>
3  #include <SPI.h>
4  #include <Wire.h>
5  #include <Adafruit_GFX.h>
6  #include <Adafruit_SSD1306.h>
7
8  void oscilloscope();
9  void Squaresignal();
10 void Sinesignal();
11 void trianglesignal();
12 void Serialget();
13
14 #define OK_buttonpin 3
15 #define Mode_buttonpin 4
16
17 #define digPin 14
18 #define DACPin A14
19 bool Ok_button_state = 0;
20 int mode_press_count = 0;
21 int mode_button_state = 0;         // current state of the button
22 const uint32_t ok_debounce_delay_ms = 200;  // Small delay for debounce
23 const uint32_t mode_debounce_delay_ms = 400;
24
25 void OK_buttonpress_ISR();                    // Interrupt function
26 void MODE_buttonpress_ISR();
27 int serialreaded;
28 int s_command  = 0;                          //variable for serial input
29 int serial_input = 0;
30 Adafruit_SSD1306 display(-1);                 //reset pin not valid
31
32 ////// For sine wave/////////
33 int yaxis = 32;                               // horizontal center of 64 / 2 = 32
34 int Radius = 30;                               // radius of circle
35 const float Pi = 3.14159265359;                // Pi
36
37 //states/////
38 enum states {
39   Welcome,
40   Menu,
41   Osciloscope,
42   Functiongenerator,
43   LogicAnalyser,
44   SinWave,
45   SquareWave,
46   TriangularWave,
47 };
48 states currentState = Welcome;                              //Welcome screen set at
   beginning
49
50 void setup()
51 {
52    // initialize the button pins as an input:
53
54    pinMode(OK_buttonpin, INPUT_PULLUP); // Enabling internal pullup resistor for
   button
55    pinMode(Mode_buttonpin, INPUT_PULLUP);
56    Serial.begin(115200);
57    analogWriteRes(10);                           //Configure DAC resolution
58   //delay(3000);
```

```
59
60    // initialize with the I2C addr 0x3C
61    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
62    attachInterrupt(digitalPinToInterrupt(OK_buttonpin), OK_buttonpress_ISR, FALLING);
  // OK button press enable interrupt function 'buttonpress_ISR'
63    attachInterrupt(digitalPinToInterrupt(Mode_buttonpin), MODE_buttonpress_ISR,
  FALLING); // OK button press enable interrupt function 'buttonpress_ISR'
64  }
65
66
67  void loop()
68  {
69  Serialget() ;                                    //function to reads serail data
70
71   switch (currentState) {                           //states and events
72
73  case Welcome:
74    if (Ok_button_state || s_command == 1 )
75     {
76         currentState = Menu;
77     }
78      display.clearDisplay();
79      display.setTextSize(2);
80      display.setTextColor(WHITE);
81      display.setCursor(20,2);
82      display.println("WELCOME");
83      display.setTextSize(1);
84      display.setCursor(0,45);
85      display.println("Press OK button to start");
86      display.display();
87  Serial.println(Ok_button_state);
88
89  //instructions for serial operations//
90    Serial.println("Modes");
91    Serial.println("Enter 8 for Welcome");
92    Serial.println("Enter 1 for Oscilloscope");
93    Serial.println("Enter 2 for Function Generator");
94    Serial.println("Enter 3 for Logic Analyzer");
95    Serial.println("Function Generator Modes");
96    Serial.println("Enter 4 for Sine Wave");
97    Serial.println("Enter 5 for Square Wave");
98    Serial.println("Enter 6 for Triangular Wave");
99    Serial.println("Enter 8 for BACK");
100
101   //Ok_button_state = 0;
102 break;
103
104 case Menu:
105
106 display.clearDisplay();
107 display.setTextSize(1);
108 display.setTextColor(WHITE);
109 display.setCursor(0,0);
110 display.println("Menu");
111 display.setCursor(10,20);
112 display.print("Osciloscope");
113 display.setCursor(10,30);
114 display.print("Function Generator");
115 display.setCursor(10,40);
116 display.print("Logic Analyser");
```

```
117 display.setCursor(10,50);
118 display.print("Go Back");
119 display.display();
120
121   if (mode_press_count == 1 || s_command == 1)             //shows the selection
    according to the mode press count
122   {
123     display.setTextSize(1);
124     display.setCursor(0,20);
125     display.print("*");
126     display.display();
127     if(Ok_button_state || s_command == 2)                          //if ok pressed or
    serial command 1 then enter to oscilloscope state
128       {
129       currentState = Osciloscope;
130       Ok_button_state = 0;
131       }
132
133   }
134   if (mode_press_count == 2  || s_command == 2)
135   {
136     display.setCursor(0,30);
137     display.print("*");
138     display.display();
139     if(Ok_button_state || s_command == 2)                      //when satisfies
    enterr  to Function generator
140       {
141       currentState = Functiongenerator;
142       Ok_button_state = 0;
143       }
144   }
145     if (mode_press_count==3  || s_command ==3)
146   {
147     display.setCursor(0,40);
148     display.print("*");
149     display.display();
150     if(Ok_button_state  || s_command ==3)
151       {
152       currentState = LogicAnalyser;                       // when satisfies enter
    into Logic analyser
153       Ok_button_state = 0;
154       }
155   }
156     if (mode_press_count==4  || s_command ==8)
157   {
158     display.setCursor(0,50);
159     display.print("*");
160     display.display();
161     if(Ok_button_state  || s_command ==8)
162       {
163       currentState = Welcome;                          // when satisfies enter into
    welcome screen
164       Ok_button_state = 0;
165       }
166   }
167   Ok_button_state = 0;
168
169  break;
170
171 case Osciloscope:                                          // Oscilloscope mode
```

```
172 oscilloscope();
173 if(Ok_button_state  || s_command ==8)
174       {
175       currentState = Welcome;                          // when satisfies enter into
    welcome screen
176       Ok_button_state = 0;
177       }
178  break;
179
180 case Functiongenerator:
     //Functiongenerator mode
181 display.clearDisplay();
182 display.setTextSize(1);
183 display.setTextColor(WHITE);
184 display.setCursor(0,0);
185 display.println("Function Generator");
186 display.setCursor(10,20);
187 display.print("Sine Wave");
188 display.setCursor(10,30);
189 display.print("Square Wave");
190 display.setCursor(10,40);
191 display.print("Triangular Wave");
192 display.setCursor(10,50);
193 display.print("Go Back");
194 display.display();
195
196
197   if (mode_press_count==1 || s_command ==4)         //Function selection for
    Function generator
198   {
199     display.setTextSize(1);
200     display.setCursor(0,20);
201     display.print("*");
202     display.display();
203     if(Ok_button_state || s_command ==4)
204       {
205       currentState = SinWave;                                 //sinewave selected if
    Ok pressed or serial command 4
206       Ok_button_state = 0;
207       }
208
209   }
210   if (mode_press_count==2  || s_command ==5)
211   {                                                          //Squarewave selected if
    Ok pressed or serial command 5
212     display.setCursor(0,30);
213     display.print("*");
214     display.display();
215     if(Ok_button_state  || s_command ==5)
216       {
217       currentState = SquareWave;
218       Ok_button_state = 0;
219       }
220   }
221     if (mode_press_count==3  || s_command ==6)
222   {                                                        //Triangularwave selected
    if Ok pressed or serial command 6
223     display.setCursor(0,40);
224     display.print("*");
225     display.display();
```

```
226     if(Ok_button_state  || s_command ==6)
227       {
228       currentState = TriangularWave;
229       Ok_button_state = 0;
230       }
231   }
232   if (mode_press_count==4  || s_command ==8)
233   {
234     display.setCursor(0,50);
235     display.print("*");
236     display.display();
237     if(Ok_button_state  || s_command ==8)
238       {
239       currentState = Welcome;                         // when satisfies enter into
welcome screen
240       Ok_button_state = 0;
241       }
242   }
243 break;
244
245 case LogicAnalyser:                                     //Logic analyser reads the
serial input numbers and convert it into Binary code
246     display.clearDisplay();
247     Serial.println("Data in BIN :");
248     Serial.println(s_command, BIN);                     //Serial display of BIN
values
249     display.setCursor(0,0);
250   display.print("Logic Analyser");
251   display.setCursor(0,30);
252   display.println("Data in BIN :");
253     display.println(s_command,BIN);                     //Oled display of BIN values
254   display.println("Data in HEX :");
255     display.println(s_command,HEX);
256     display.setTextSize(1);
257     display.display();
258     if(Ok_button_state  || s_command ==8)
259       {
260       currentState = Welcome;                         // when satisfies enter into
welcome screen
261       Ok_button_state = 0;
262       }
263 break;
264
265 case SinWave:
266   display.clearDisplay();                                   // clears display
267   Sinesignal();                                             //Sine wave
generating function
268   if(Ok_button_state  || s_command ==8)
269       {
270       currentState = Welcome;                         // when satisfies enter into
welcome screen
271       Ok_button_state = 0;
272       }
273 break;
274
275 case SquareWave:
276   Squaresignal();
277   if(Ok_button_state  || s_command ==8)
278       {
```

```
279       currentState = Welcome;                          // when satisfies enter into
welcome screen
280       Ok_button_state = 0;
281       }
282 break;
283
284 case TriangularWave:
285 trianglesignal();
286 if(Ok_button_state  || s_command ==8)
287       {
288       currentState = Welcome;                          // when satisfies enter into
welcome screen
289       Ok_button_state = 0;
290       }
291 break;
292 }}
293 void Serialget()                                              //Serial
communication function
294 {
295
296   if( Serial.available())
297   {
298      char serial_read = Serial.read();                          //serial read and
store it
299     if(serial_read >= '0' && serial_read <= '9')
300     {
301     serial_input = (serial_input * 10) + (serial_read - '0');   //convert ASCI code
302     }
303     else if (serial_read == 10)                                 // is the character
the newline character
304     {
305       s_command = serial_input;                                // stored to Serial
command
306       Serial.println(s_command); //error check
307     //serialreaded = s_command;
308       serial_input = 0;                                         // reset val to 0
ready for the next sequence of digits
309     }
310    }
311 }
312
313 void Squaresignal()
314 {
315   display.clearDisplay();                                     // clears display
316   display.drawLine(0,30,120,30,WHITE);                        // draws grid
horizontal center
317   display.drawLine(0,0,0,64,WHITE);                          // draws grid vertical
first
318
319
320   display.setCursor(95,34);                                   // X-axis labels
321   display.print("t(ms)");
322   display.setCursor(0,0);
323   display.print("v(mv)");                                     // Y-axis labels
324   display.display();
325   static int y = 0 ;
326
327   for (int i=0; i<120; i++)                                   // draws 120 pixels
per loop I stores X Axis value and y stores Yaxis value
328   {
```

```
329
330
331
332
333
334       if(i<60)                                                  //Y plotted high till
    half timeperiod
335       {
336         y = 10;
337       }
338      /*
339       if(i==60)
340
341       {
342       display.drawLine(30,10,30,54,WHITE);                    //Drop line
343       }
344        if(i>30)                                                  //Y plotted low till
    next half timeperiod
345       {
346         if(i<60)
347         {y = 54;}
348       }
349       */
350       if(i==60)
351       {
352        display.drawLine(60,10,60,54,WHITE);                    //Drop line
353       }
354       if(i>60)
355       {
356       if(i<120)
357         {y = 54;}
358       }
359     if(i==120)
360       {
361        display.drawLine(90,10,90,54,WHITE);
362       }
363       /*if(i>90)
364       {
365         y=54;
366       }
367       if(i==120)
368       {
369        display.drawLine(120,54,120,10,WHITE);
370       }
371       */
372   //Serial.println(y);                                          // for serial plotting
373     display.drawPixel(i,y,WHITE);                              // draws each pixel
    for Square wave
374     display.display();                                         // displays new
    screen information
375     analogWrite(DACPin, y);                                    // send signal to
    DAC
376   }
377 }
378
379 void Sinesignal()
380 {
381
382   display.clearDisplay();                                      // clears display
```

```
383   display.drawLine(0,30,120,30,WHITE);                          // draws grid
horizontal center
384   display.drawLine(0,0,0,64,WHITE);                            // draws grid vertical
first
385   display.setCursor(95,34);
386   display.print("t(ms)");
387   display.setCursor(0,0);
388   display.print("v(mv)");
389   display.display();
390   for (int i=0; i<120; i++)                                     // draws 120 pixels
per loop
391    {
392     float Angle = i*3 ;                                         // 120 X 3 = 360°
393     int a = (yaxis - (sin(Angle * (Pi / 180)) * Radius));
394     //Serial.println(a);                                         // for serial
plotting
395     display.drawPixel(i,a,WHITE);                               // draws each pixel
for Sine wave
396     display.display();                                          // displays new
screen information
397     analogWrite(DACPin,a);                                      // send signal to DAC

398   }
399 }
400
401 void trianglesignal()
402 {
403
404   display.clearDisplay();                                       // clears display
405   display.drawLine(0,30,120,30,WHITE);                          // draws grid
horizontal center
406   display.drawLine(0,0,0,64,WHITE);                            // draws grid vertical
first
407
408
409   display.setCursor(95,34);                                     // axis labels
410   display.print("t(ms)");
411   display.setCursor(0,0);
412   display.print("v(mv)");
413   display.display();
414   static int y = 0 ;
415
416   for (int i=0; i<120; i++)                                    // draws 120 pixels per
loop
417   {
418
419       if(i<=30)                                                // Y increasing till
1/4 timeperiod
420       {
421         y = i;
422       }
423       if(i>30)
424       {
425         if(i<60)
426         {y--;}                                                  //Y decreasing next
1/4 timeperiod
427       }
428       if(i>=60)
429       {
430       if(i<90)
```

```
431         {y = i-60;}
432       }
433       if(i>90)
434       {
435         y--;
436       }
437     // Serial.println(y); //serial plotting
438     display.drawPixel(i,y,WHITE);                                   // draws each pixel
for Square wave
439     display.display();                                              // displays new
screen information
440     analogWrite(DACPin, y);                                          // send signal to
DAC
441   }
442 }
443
444 void oscilloscope()
445 {
446
447   display.clearDisplay();                                           // clears display
448   display.drawLine(0,30,120,30,WHITE);                              // draws grid
horizontal center
449   display.drawLine(0,0,0,64,WHITE);                                // draws grid vertical
first
450
451   display.setCursor(95,34);
452   display.print("t(ms)");
453   display.setCursor(0,0);
454   display.print("3.3V");
455   display.setCursor(0,32);
456   display.print("0");
457   display.display();
458
459 for(int x=0;x<120;x++)
460     {
461
462    int sensorValue = analogRead(A0);                                    // read the
input on analog pin 0:
463   float voltage = sensorValue * (3.3 / 1023.0);                         // Convert the
analog reading (which goes from 0 - 1023) to a voltage (0 - 3.3V):
464   voltage = 32-(voltage*7);                                             // y axis value:
multiplied with 7 to scale y value
465   display.drawPixel(x,voltage,WHITE);                                   // draws each
pixel
466     display.display();                                                  // displays new
screen information
467
468     }
469
470 }
471
472 void OK_buttonpress_ISR()                                                //
Interrupt function for OK button
473 {
474   static uint32_t previoustimebutton_ms = 0;                           //
Variable for debouncing
475   uint32_t currenttimebutton_ms = millis();                            //
Variable for debouncing
476    if((currenttimebutton_ms - previoustimebutton_ms) >= ok_debounce_delay_ms) //
Code inside loop only executes if buttonpress time is greater than debounce delay_ms
```

```
  { Ok_button_state = !Ok_button_state;
    previoustimebutton_ms = currenttimebutton_ms;

  }
}

void MODE_buttonpress_ISR()                                          //
Interrupt function for OK button
{
static uint32_t previoustime_MODE_button_ms = 0;                      //
button debouncing
uint32_t currenttime_MODE_button_ms = millis();                       //
button debouncing
if((currenttime_MODE_button_ms - previoustime_MODE_button_ms) >=
mode_debounce_delay_ms) // executes only if buttonpress time is greater than
debounce delay_ms
  {

      mode_press_count++;


      if(mode_press_count >4)
      {
        mode_press_count=1;
      }

  }
}


```