

仿真图

1. #include <LiquidCrystal.h>    //下载更新库
2. **const** **int** rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;//定义引脚
3. LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
5. **int** pulsePin = A0;
6. **int** blinkPin = 13;                //引脚在每次节拍时闪烁
8. **volatile** **int** BPM;
9. **volatile** **int** Signal;
10. **volatile** **int** IBI = 600;             //保存节拍之间的时间间隔
11. **volatile** boolean Pulse = **false**;     //检测到用户的实时心跳时为“真”。不是时为“假”。
12. **volatile** boolean QS = **false**;
14. **static** boolean serialVisual = **true**;
16. **volatile** **int** rate[10];                // 保存最后十个 IBI 值的数组
17. **volatile** unsigned **long** sampleCounter = 0;          // 用于确定脉冲时序
18. **volatile** unsigned **long** lastBeatTime = 0;
19. **volatile** **int** P = 512;                     //脉冲峰值
20. **volatile** **int** T = 512;                    // 脉冲波谷
21. **volatile** **int** thresh = 525;
22. **volatile** **int** amp = 100;
23. **volatile** boolean firstBeat = **true**;
24. **volatile** boolean secondBeat = **false**;
25. **void** setup()
26. {
27. pinMode(blinkPin,OUTPUT);    //屏幕随心跳闪烁
28. Serial.begin(115200);       //波特率选项
29. interruptSetup();           //设置为每2mS读取一次脉冲传感器信号
31. lcd.begin(16, 2);    //初始化LCD1602
32. lcd.clear();    //液晶清屏
33. }
35. **void** loop()
36. {
37. serialOutput();
38. **if** (QS == **true**)
39. {
40. serialOutputWhenBeatHappens();
41. QS = **false**;        // 下次重置量化标志
42. }
43. delay(20);
44. }
46. **void** interruptSetup()
47. {
48. TCCR2A = 0x02;    //禁用数字引脚3和11上的PWM，并进入CTC模式
49. TCCR2B = 0x06;     //不要强制比较，256预分频器
50. OCR2A = 0X7C;     //对于500Hz采样率，将计数顶部设置为124
51. TIMSK2 = 0x02;    //在定时器2和OCR2A之间匹配时启用中断
52. sei();              //确保启用了全局中断
53. }
55. **void** serialOutput()
56. {
57. **if** (serialVisual == **true**)
58. {
59. arduinoSerialMonitorVisual('-', Signal);
60. }
61. **else**
62. {
63. sendDataToSerial('S', Signal);
64. }
65. }
67. **void** serialOutputWhenBeatHappens()
68. {
69. **if** (serialVisual == **true**)
70. {
71. Serial.print(" Heart-Beat Found ");
72. Serial.print("BPM: ");
73. Serial.println(BPM);
74. lcd.print("Heart-Beat Found ");  //LCD显示内容
75. lcd.setCursor(1,1);  //设置液晶开始显示的指针位置，1列1行
76. lcd.print("BPM: ");
77. lcd.setCursor(5,1);
78. lcd.print(BPM);
79. delay(300);    //延时300ms
80. lcd.clear();
81. }
82. **else**
83. {
84. sendDataToSerial('B',BPM);  //发送带有“B”前缀的心率
85. sendDataToSerial('Q',IBI);
86. }
87. }
89. **void** arduinoSerialMonitorVisual(**char** symbol, **int** data )
90. {
91. **const** **int** sensorMin = 0;    //定义整形常量Minn
92. **const** **int** sensorMax = 1024;
93. **int** sensorReading = data;
94. **int** range = map(sensorReading, sensorMin, sensorMax, 0, 11);
95. }
97. **void** sendDataToSerial(**char** symbol, **int** data )
98. {
99. Serial.print(symbol);
100. Serial.println(data);
101. }
103. ISR(TIMER2\_COMPA\_vect)     //计时器2计数为124时触发
104. {
105. cli();               //执行此操作时禁用中断
106. Signal = analogRead(pulsePin);        //读取脉冲传感器
107. sampleCounter += 2;
108. **int** N = sampleCounter - lastBeatTime;      //监控自上次节拍以来的时间，以避免干扰
110. **if**(Signal < thresh && N > (IBI/5)\*3)       //等待最后一次IBI的3/5以避免二次干扰
111. {
112. **if** (Signal < T)
113. {
114. T = Signal;  // 跟踪脉搏波的最低点
115. }
116. }
118. **if**(Signal > thresh && Signal > P)     //脱粒状态有助于避免干扰
119. {
120. P = Signal;        // 跟踪脉搏波的最高点
121. }

124. **if** (N > 250)   //读数大于250
125. {
126. **if** ( (Signal > thresh) && (Pulse == **false**) && (N > (IBI/5)\*3) )
127. {        //当有脉冲时，设置脉冲标志
128. Pulse = **true**;
129. digitalWrite(blinkPin,HIGH);          //打开针脚13 LED
130. IBI = sampleCounter - lastBeatTime;
131. lastBeatTime = sampleCounter;
133. **if**(secondBeat)
134. {
135. secondBeat = **false**;                 // 清除 secondBeat 标志
136. **for**(**int** i=0; i<=9; i++)
137. {
138. rate[i] = IBI;
139. }
140. }
142. **if**(firstBeat)
143. {
144. firstBeat = **false**;                    // 清除 firstBeat 标志
145. secondBeat = **true**;
146. sei();
147. **return**;
148. }
150. word runningTotal = 0;                 // 清除 runningTotal 变量
152. **for**(**int** i=0; i<=8; i++)
153. {
154. rate[i] = rate[i+1];                 // 删除最旧的 IBI 值
155. runningTotal += rate[i];               // 将 9 个最旧的 IBI 值相加
156. }
158. rate[9] = IBI;                           // 将最新的 IBI 添加到速率数组中
159. runningTotal += rate[9];
160. runningTotal /= 10;
161. BPM = 60000/runningTotal;              // 一分钟心跳次数
162. QS = **true**;                               // 设置 "量化自我 "标志
163. }
164. }
166. **if** (Signal < thresh && Pulse == **true**)
167. {   //当数值下降时，节拍结束
168. digitalWrite(blinkPin,LOW);            // 关掉第13针LED
169. Pulse = **false**;
170. amp = P - T;
171. thresh = amp/2 + T;                    //将阈值设置为振幅的50%
172. P = thresh;                            // 重置这些以备下次使用
173. T = thresh;
174. }
176. **if** (N > 2500)
177. {
178. thresh = 512;             //设置默认阈值
179. P = 512;
180. T = 512;
181. lastBeatTime = sampleCounter;        //将lastBeatTime更新到最新状态
182. firstBeat = **true**;
183. secondBeat = **false**;
184. }
186. sei();             //完成后启用中断
187. }