**MCES PRODUCTATHON**

**TITLE: BOMB DIFFUSION SIMULATOR**

**Description of the Product:**

The "Bomb Diffusion Simulator" is a mini project designed to simulate the process of diffusing a bomb using various electronic components and a microcontroller, in this case, the LPC2148. Here's a detailed description of how the project works:

Hardware Components Used:

1. LPC2148 Microcontroller: This serves as the brain of the system and controls all the other components. It handles the logic for bomb diffusion, password entry, countdown timer, and interaction with peripherals.
2. Proximity Sensor: The proximity sensor is used to detect if a person has approached the bomb to diffuse it.
3. LCD Display: An LCD screen is used to provide visual feedback to the user. It initially displays the message "Enter pass:" and later shows either "Bomb Diffused" or "Bomb Exploded" based on the outcome.
4. 4x4 Matrix Keyboard: This keyboard is used for password input. It consists of a 4x4 grid of keys, allowing users to enter a numerical password.
5. Seven-Segment Display: A seven-segment display is used to show the countdown timer. It displays the remaining time the user must diffuse the bomb.

Working of the Product:

1. Initialization: When the system is powered on or reset, the microcontroller initializes all the components and prepares the display and countdown timer.
2. Display Initialization: The LCD display is set up to show the initial message, "Enter pass:"
3. Countdown Timer and Proximity Sensor: The seven-segment display is initialized with a countdown timer set to 10 seconds. This timer starts counting down as soon as the system is ready. This timer runs in the background with the help of Interrupts. While in the foreground, the proximity sensor is continuously detecting for a person to come and diffuse the bomb.
4. Countdown Stop: If the proximity sensor detects a person, the countdown timer is stopped.
5. Password Entry: The user is then prompted to enter a password using the 4x4 matrix keyboard. The entered password is compared to a predefined correct password.
6. Outcome Display: Depending on whether the correct password was entered before the proximity sensor detected a person, the LCD display will show one of two messages:

If the password is correct and the proximity sensor detects a person, it will display "Bomb diffused!" If the password is incorrect and the proximity sensor detects a person, it will display "Bomb exploded!"

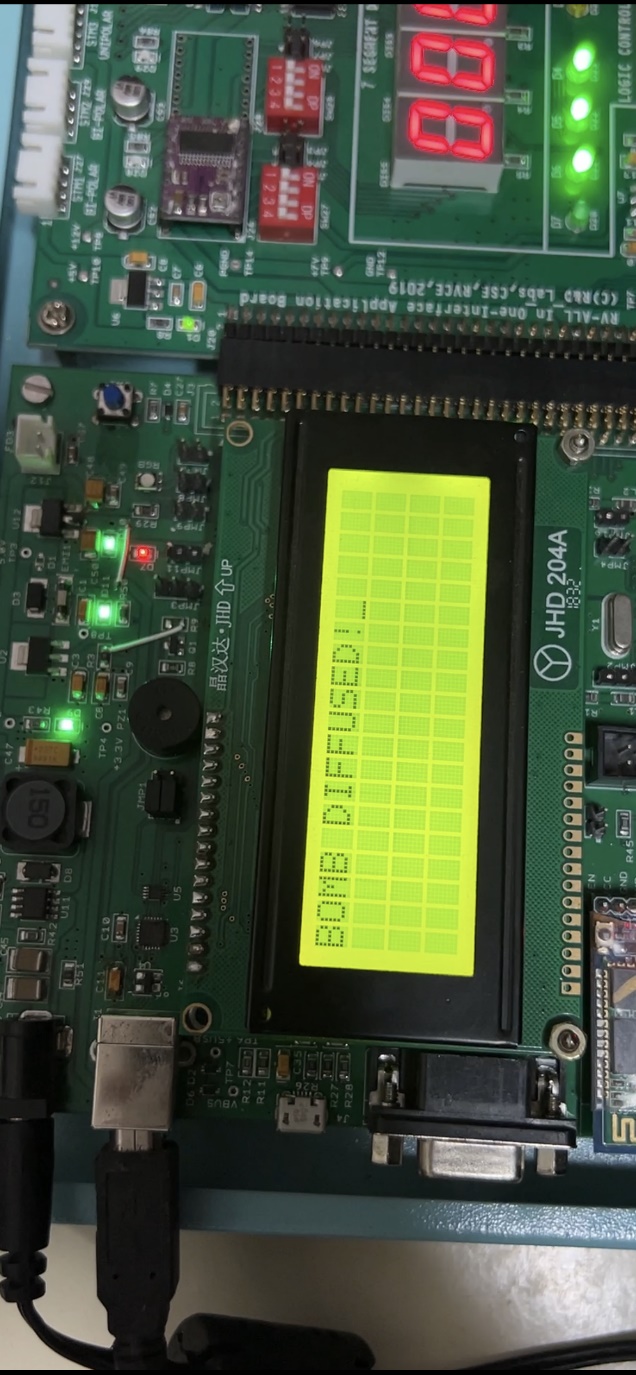
**Product Images:**

A green electronic board with red numbers and a yellow screen

Description automatically generated

A hand holding a small yellow object

Description automatically generated



**Source Code:**

1. #include <lpc214x.h>

2. #include <string.h>

3. #define CLEAR (IO0SET = 0x000F0000)

4. #define SET1 IO0CLR = 1 << 16

5. #define SET2 IO0CLR = 1 << 17

6. #define SET3 IO0CLR = 1 << 18

7. #define SET4 IO0CLR = 1 << 19

8. #define C0(IO1PIN & 1 << 19)

9. #define C1(IO1PIN & 1 << 18)

10. #define C2(IO1PIN & 1 << 17)

11. #define C3(IO1PIN & 1 << 16)

12. #define LED\_ON IO0CLR |= 1 << 31

13. #define LED\_OFF IO0SET |= 1 << 31

14. #define Check0 if (C0 == 0) { col = 0; break; }

15. #define Check1 if (C1 == 0) { col = 1; break; }

16. #define Check2 if (C2 == 0) { col = 2; break; }

17. #define Check3 if (C3 == 0) { col = 3; break; }

18. #define ENTER '0'

19. #define WAIT while (C0 == 0 || C1 == 0 || C2 == 0 || C3 == 0);

20. #define RS\_ON (IO0SET = 1 U << 20)

21. #define RS\_OFF (IO0CLR = 1 U << 20)

22. #define EN\_ON (IO1SET = 1 U << 25)

23. #define EN\_OFF (IO1CLR = 1 U << 25)

24. unsigned int x = 0;

25. long int count = 10;

26. char buf[5];

27. char ch, keys[5], password[5] = "1234";

28. unsigned char len = 0;

29. unsigned int i = 0;

30.

31. void delay(int x) {

32. for (int i = 0; i < x; i++)

33. for (int y = 0; y < 10000; y++)

34. ;

35. }

36. static void delay\_us(unsigned int count) {

37. unsigned int j = 0, i = 0;

38. for (j = 0; j < count; j++) {

39. for (i = 0; i < 10; i++)

40. ;

41. }

42. }

43. static void LCD\_SendCmdSignals(void) {

44. RS\_OFF; EN\_ON;

45. delay\_us(100);

46. EN\_OFF;

47. }

48. static void LCD\_SendDataSignals(void) {

49. RS\_ON; EN\_ON;

50. delay\_us(100);

51. EN\_OFF;

52. }

53. static void LCD\_SendHigherNibble(unsigned char dataByte) {

54. // send the D7,6,5,D4(uppernibble) to P0.16 to P0.19

55. IO0CLR = 0X000F0000;

56. IO0SET = ((dataByte >> 4) & 0x0f) << 16;

57. }

58. static void LCD\_CmdWrite(unsigned char cmdByte) {

59. LCD\_SendHigherNibble(cmdByte);

60. LCD\_SendCmdSignals();

61. cmdByte = cmdByte << 4;

62. LCD\_SendHigherNibble(cmdByte);

63. LCD\_SendCmdSignals();

64. }

65. static void LCD\_DataWrite(unsigned char dataByte) {

66. LCD\_SendHigherNibble(dataByte);

67. LCD\_SendDataSignals();

68. dataByte = dataByte << 4;

69. LCD\_SendHigherNibble(dataByte);

70. LCD\_SendDataSignals();

71. }

72. static void LCD\_Reset(void) {

73. /\* LCD reset sequence for 4-bit mode\*/

74. LCD\_SendHigherNibble(0x30);

75. LCD\_SendCmdSignals();

76. delay(100);

77. LCD\_SendHigherNibble(0x30);

78. LCD\_SendCmdSignals();

79. delay\_us(200);

80. LCD\_SendHigherNibble(0x30);

81. LCD\_SendCmdSignals();

82. delay\_us(200);

83. LCD\_SendHigherNibble(0x20);

84. LCD\_SendCmdSignals();

85. delay\_us(200);

86. }

87. static void LCD\_Init(void) {

88. delay(100);

89. LCD\_Reset();

90. LCD\_CmdWrite(0x28u); // Initialize the LCD for 4-bit 5x7 matrix type

91. LCD\_CmdWrite(0x0Eu); // Display ON cursor ON

92. LCD\_CmdWrite(0x01u); // Clear the LCD

93. LCD\_CmdWrite(0x80u); // go to First line First Position

94. }

95. void LCD\_DisplayString(const char \* ptr\_string) {

96. while (( \* ptr\_string) != 0)

97. LCD\_DataWrite( \* ptr\_string++);

98. }

99. unsigned char getAlphaCode(unsigned char alphachar) {

100. switch (alphachar) {

101. case 'f': return 0x8e;

102. case 'i': return 0xf9;

103. case 'r': return 0xce;

104. case 'e': return 0x86;

105. case 'h': return 0x89;

106. case 'l': return 0xc7;

107. case 'p': return 0x8c;

108. case ' ': return 0xff;

109. case '1': return 0xf9;

110. case '2': return 0xa4;

111. case '3': return 0xb0;

112. case '4': return 0x99;

113. case '5': return 0x92;

114. case '6': return 0x82;

115. case '7': return 0xf8;

116. case '8': return 0x80;

117. case '9': return 0x90;

118. case '0': return 0xc0;

119. default: break;

120. }

121. return 0xff;

122. }

123. void alphadisp7SEG(char \* buf) {

124. unsigned char i, j;

125. unsigned char seg7\_data, temp = 0;

126. for (i = 0; i < 5; i++) {

127. seg7\_data = getAlphaCode( \* (buf + i));

128.

129. for (j = 0; j < 8; j++) {

130. temp = seg7\_data & 0x80;

131.

132. if (temp == 0x80)

133. IOSET0 |= 1 << 19;

134. else

135. IOCLR0 |= 1 << 19;

136.

137. IOSET0 |= 1 << 20;

138. delay(1);

139. IOCLR0 |= 1 << 20;

140. seg7\_data = seg7\_data << 1;

141. }

142. }

143.

144. IOSET0 |= 1 << 30;

145. delay(1);

146. IOCLR0 |= 1 << 30;

147. return;

148. }

149. \_\_irq void Timer0\_ISR(void) // an ISR program

150. {

151. // 7seg logic

152. sprintf(buf, "%05lu", count);

153. alphadisp7SEG( & buf[0]);

154. count--;

155. if (count == 0) {

156. count = 10;

157. }

158. delay(600);

159. T0IR = 0x01; // clear match0 interrupt, and get ready for the next interrupt

160. VICVectAddr = 0x00000000; // End of interrupt

161. }

162. char getKey() {

163. unsigned char lookup\_table[4][4] = {

164. {'0','1','2','3'},

165. {'4','5','6','7'},

166. {'8','9','a','b'},

167. {'c','d','e','f'}

168. };

169. unsigned int row = 0, col = 0;

170. do {

171. row = 0; CLEAR; SET1;

172. Check0; Check1; Check2; Check3;

173.

174. row = 1; CLEAR; SET2;

175. Check0; Check1; Check2; Check3;

176.

177. row = 2; CLEAR; SET3;

178. Check0; Check1; Check2; Check3;

179.

180. row = 3; CLEAR; SET4;

181. Check0; Check1; Check2; Check3;

182. } while (1);

183.

184. delay(50); // debouncing

185. WAIT;

186. delay(50); // debouncing

187.

188. return lookup\_table[row][col];

189. }

190.

191. int readSensor(int sen\_no) {

192. int result = 0;

193. IO1DIR |= 1 << 24;

194. IO1CLR |= 1 << 24; // enable sensor logic

195. switch (sen\_no) {

196. case 1:

197. result = IO1PIN & (1 << 22);

198. break;

199. case 2:

200. result = IO1PIN & (1 << 23);

201. break;

202. default:

203. result = 0;

204. }

205. IO1SET = 1 << 24;

206. return result;

207. }

208.

209. int main() {

210. int result = 0;

211. IO1DIR |= 1 U << 25;

212. IO0DIR |= 1 U << 31 | 1 U << 19 | 1 U << 20 | 1 U << 30 | 0x00FF0000;

213.

214. LED\_OFF;

215.

216. LCD\_Reset();

217. LCD\_Init();

218. delay(100);

219. LCD\_CmdWrite(0x80);

220. LCD\_CmdWrite(0x80);

221.

222. T0TCR = 0x00; // stop the timer, to initialize different registers

223. T0MCR = 0x0003; // Enable Interrupt and reset timer after match

224. T0TC = 0x00; // make TC = 0

225. T0MR0 = 1500000; // generates 1s

226.

227. // load interrupt related registers , assigning Timer0 to IRQ slot 4

228. VICVectAddr4 = (unsigned long) Timer0\_ISR; // set the timer ISR vector address

229. VICVectCntl4 = 0x0000024; // set the channel

230. VICIntEnable = 0x0000010; // enable the timer0 interrupt

231. T0TCR = 0x01; // start the timer

232.

233. LCD\_DisplayString("Enter Pass: ");

234.

235. while (1) {

236. while ((result = readSensor(2)) != 0) {

237. char ch;

238. do {

239. i = 0;

240. VICIntEnable = 0x00000000;

241. T0TCR = 0x00; // start the timerdelay(10);

242. IO0SET |= 1 U << 16 | 1 U << 17 | 1 U << 18 | 1 U << 19;

243. while (1) {

244. if ((ch = getKey()) == ENTER)

245. break;

246. keys[i++] = ch;

247. }

248. keys[i] = '\0';

249.

250. if ((strcmp(keys, password)) == 0) {

251. LCD\_CmdWrite(0x80); // go to First line First Position

252. LCD\_DisplayString("BOMB DIFFUSED!");

253. LED\_ON;

254. delay(2000);

255. return 0;

256. }

257. LCD\_CmdWrite(0x80);

258. LCD\_DisplayString("BOMB EXPLODED!");

259.

260. } while (1);

261. }

262. }

263. return 0;

264. }