Final Project - Math Quizer

Introduction to Embedded Systems - University of Nebraska

Dat Nguyen

Contents

1 Introduction

This report will provide detail the Arduino Math Quizer Program

2 Program Description

Video demo: https://youtu.be/2AjNVgh04Rc

1 + 2. The program first enters INTRO state and greeting message is printed. Programs ask for number of question and validate if the user input is falled in 5 - 25 range. If conditions don't meet, then program reprompt the user

3. A randomize math question function is used to generate. Seeding is based on the timestamp provided by the builtin millis() function. The seed is updated every loop cycle. The random math question is implemented as follow:

```
typedef struct{
  char mathOperator; // * or /
  int num1;
  int num2;
  int expectedResult;
} MathQuestion;

MathQuestion generateRandomMathQuestion() {
  MathQuestion question;
  question.mathOperator = (rand() % 2) == 0 ? '*' : '/';
  int num1 = (rand() % 10) + 1, num2 = (rand() % 10) + 2;
  if (question.mathOperator == '*') {
    question.num1 = num1;
    question.num2 = num2;
    question.expectedResult = num1 * num2;
  } else {
    question.num1 = num1 * num2;
    question.num2 = num1;
    question.expectedResult = num2;
  }
  return question;
}
```

For division, the question can be modeled as a / b = c. b and c will be randomly generated to be fell within 1-10 range. a is solved by multiply b with c.

4. After answering each question, statistic like current question index and time per question is printed. At the end, number of correct questions and average time per question are also reported.

5. A debounced button pressed can be triggered by user at anytime to restart the quiz as shown in the demo

6. Upon incorrectly answered question, a red LED will blink for several time.

7. In this hardware implementation, a 220 Ohm was used as the limiting resistor. Following Ohm's law, I = V/R, 22.7mA of current will flow through the LED upon triggering. This is within the spec of the uC part tolerance. Figure 1 display the circuit in more detail

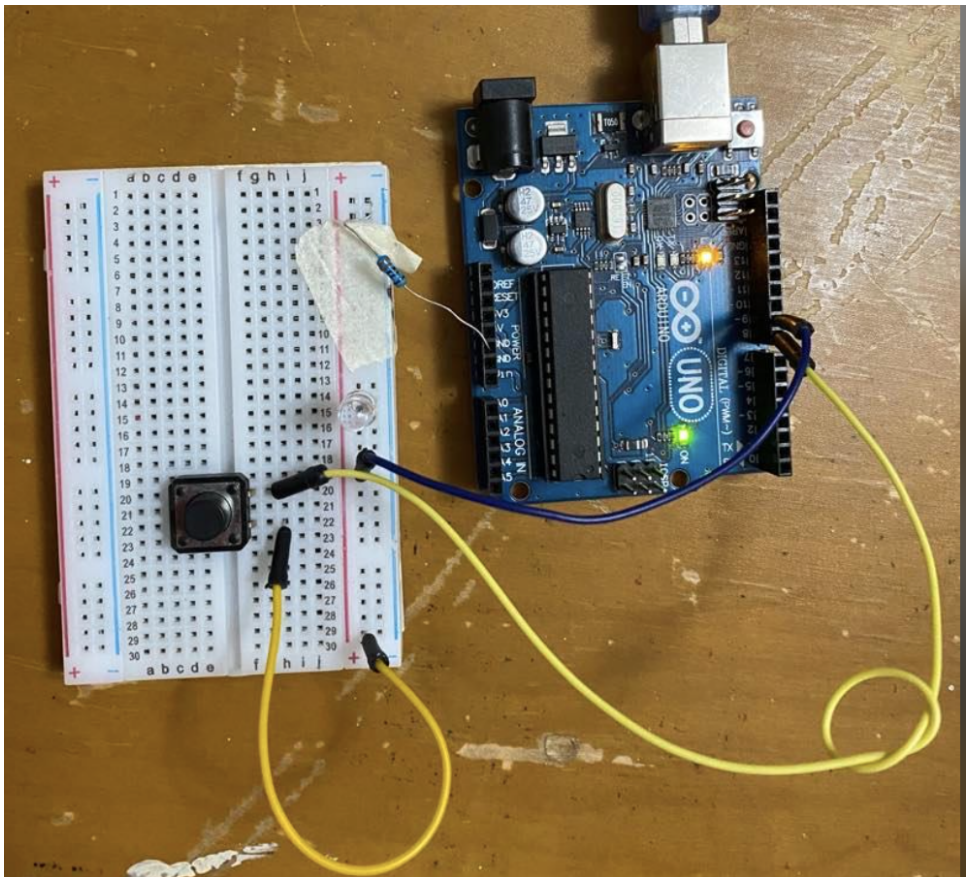

Figure 1: Math Quizer Arduino Circuit

```
==============================
Welcome to quizer version 1.0.0
How many number of questions would you like to quiz on? (5 <= numQuestions <= 25)
>>> 5
This many question: 5
What is 50 / 10? (1 out of 5 questions)
>>> 5
CORRECT: 5 - time: 9444ms
Current score: 1/5
What is 8 * 8? (2 out of 5 questions)
>>> 1
WRONG: 1 - time: 1936ms
Current score: 1/5
What is 35 / 5? (3 out of 5 questions)
>>> 1
WRONG: 1 - time: 1096ms
Current score: 1/5
What is 10 * 8? (4 out of 5 questions)
>>> 80
CORRECT: 80 - time: 3123ms
Current score: 2/5
What is 9 * 7? (5 out of 5 questions)
>>> 1
WRONG: 1 - time: 1786ms
Current score: 2/5
Congrats!! Your score: 2/5
Restarting...
```

Figure 2: Math Quizer Example Program

3 Appendix

3.1 main.c

```c
#include <stdint.h>;
#include "expriments.h"

int main(void) {
  init();
  mainProgram();
  return 0;
}
```

3.2 expriments.h

```c
#ifndef EXPERIMENTS_H
#define EXPERIMENTS_H
#include <stdio.h>

typedef enum {
  OVERFLOW = 0,
  FILLED,
  READING
} uart_state;


typedef enum {
  INTRO = 0,
  ASK_NUM_QUESTION,
  WAIT_NUM_QUESTION,
  ASK_QUESTION,
  WAIT_FOR_ANSWER,
  FINISH_QUIZ,
  RESTART_QUIZ
} quiz_state;


typedef struct{
  char mathOperator; // * or /
  int num1;
  int num2;
  int expectedResult;
} MathQuestion;

void mySerialBegin(uint32_t baudrate);
uint8_t serialRead();
boolean serialAvailable();
void mySerialWriteOne(uint8_t data);
void mySerialWrite(uint8_t * msg);
void flushRxUartBuffer();

void myHardDelay(uint32_t ms);

uint8_t readInputPB0();
void configurePB0ButtonAndPB1();
int debouncePB0();
void turnOnPB1LED();
void turnOffPB1LED();

MathQuestion generateRandomMathQuestion();

void mainProgram();

#endif
```

3.3 expriments.cpp

```
1
2  #include <Arduino.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include "avr/interrupt.h"
6  #include "expriments.h"
7
8  #define FOSC 16000000 // Clock speed
9  #define BAUDRATE 9600
10 #define BAUD2UBRR(baud) FOSC/16/baud-1
11
12 volatile uint8_t *pUBRR0L,
13                  *pUBRR0H,
14                  *pUCSRnA,
15                  *pUCSR0B, // USART Control and Status Register 0 B
16                  *pUCSR0C,   // USART Control and Status Register 0 C
17                  *pUDRn,
18                  *ioDDRB,
19                  *ioPORTB, // config button
20                  *ioPINB  // config button
21 ;
22
23
24
25 // Global variable
26 #define UART_BUFFER_SIZE 10
27
28 uint8_t rxUartBuffer[UART_BUFFER_SIZE+1];
29 uint16_t rxUartIndex = 0;
30
31 char msg[100];
32
33 uart_state uartState = READING;
34 quiz_state quizState = INTRO;
35
36 #define BUTTON_PRESSED     0
37 #define BUTTON_NOT_PRESSED 1
38 uint8_t pinPB0State = 0;
39
40 volatile unsigned long startQuestionTimer = 0;
41
42 void mainProgram() {
43   mySerialBegin(BAUDRATE);
44   configurePB0ButtonAndPB1();
45   uint16_t numQuestions = 0;
46   uint16_t questionIndex = 0;
47   uint16_t correctAnswers = 0;
48   MathQuestion question;
49   while(1){
50     /* Intializes random number generator */
51     srand(millis()); // seed rand() else rand() would just return repeated sequence upon
         each reboot
52     if (debouncePB0()) {
53       if (pinPB0State == BUTTON_NOT_PRESSED) {
54       } else {
55         quizState = RESTART_QUIZ;
56       }
57     }
58
59     if(serialAvailable() && (uartState != OVERFLOW || uartState != FILLED) ) {
60       if (rxUartIndex >= UART_BUFFER_SIZE) {
61         mySerialWrite("\nERROR: overflow, truncated\n");
62         uartState = OVERFLOW;
63         flushRxUartBuffer();
64       } else {
```

```
65          uint8_t ascii_char = serialRead();
66          rxUartBuffer[rxUartIndex++] = ascii_char;
67          if (ascii_char == '\n') {
68            rxUartBuffer[rxUartIndex] = '\0'; // null terminator
69            uartState = FILLED;
70            mySerialWrite(">>> ");
71            mySerialWrite((char *)rxUartBuffer);
72          }
73        }
74      }
75      if (quizState == INTRO) {
76        mySerialWrite("\n==========\n");
77        mySerialWrite("Welcome to quizer version 1.0.0\n");
78        quizState = ASK_NUM_QUESTION;
79      } else if (quizState == ASK_NUM_QUESTION) {
80        mySerialWrite("How many number of questions would you like to quiz on? (5 <= numQuestions <= 25)\n");
81        quizState = WAIT_NUM_QUESTION;
82      } else if (quizState == WAIT_NUM_QUESTION) {
83        if (uartState == FILLED) {
84          numQuestions = atoi((char*)rxUartBuffer);
85          sprintf(msg, "This many question: %d\n", numQuestions);
86          mySerialWrite((char *) msg);
87          flushRxUartBuffer();
88          if (numQuestions >= 5 && numQuestions <= 25) {
89            quizState = ASK_QUESTION;
90          } else {
91            mySerialWrite("ERROR: Make sure number of questions is (5 <= numQuestions <= 25)\n");
92            quizState = ASK_NUM_QUESTION;
93          }
94        }
95      } else if (quizState == ASK_QUESTION) {
96        question = generateRandomMathQuestion();
97        sprintf(msg, "What is %d %c %d? (%d out of %d questions)\n", question.num1, question.mathOperator, question.num2, questionIndex+1, numQuestions);
98        mySerialWrite((char *) msg);
99        startQuestionTimer = millis();
100       quizState = WAIT_FOR_ANSWER;
101     } else if (quizState == WAIT_FOR_ANSWER) {
102       if (uartState == FILLED) {
103         unsigned long totalTime = (millis() - startQuestionTimer);
104         uint16_t answer = atoi((char*)rxUartBuffer);
105         flushRxUartBuffer();
106
107         if (answer == question.expectedResult) { // correct
108           sprintf(msg, "CORRECT: %d - time: %ldms\n", answer, totalTime);
109           correctAnswers++;
110         } else {
111           for (int i=0; i<3; i++) {
112             turnOnPB1LED();
113             myHardDelay(100);
114             turnOffPB1LED();
115             myHardDelay(100);
116           }
117           sprintf(msg, "WRONG: %d - time: %ldms\n", answer, totalTime);
118         }
119         mySerialWrite((char *) msg);
120         sprintf(msg, "Current score: %d/%d\n", correctAnswers, numQuestions);
121         mySerialWrite(msg);
122         questionIndex++;
123         if (questionIndex == numQuestions) {
124           quizState = FINISH_QUIZ;
125         } else {
126           quizState = ASK_QUESTION;
127         }
128       }
129     } else if (quizState == FINISH_QUIZ) {
```

```
130         sprintf(msg, "Congrats!! Your score: %d/%d\n", correctAnswers, numQuestions);
131         mySerialWrite(msg);
132         quizState = RESTART_QUIZ;
133       } else if (quizState == RESTART_QUIZ) {
134         mySerialWrite("Restarting...\n\n");
135         numQuestions = 0;
136         questionIndex = 0;
137         correctAnswers = 0;
138         quizState = INTRO;
139       }
140     }
141   }
142   /************************** Configure PB0 Button to restart quiz and PB1 as output to
          turn on LED *************/
143   void configurePB0ButtonAndPB1() {
144     ioPINB  = (uint8_t *) 0x23;
145     ioDDRB = (uint8_t *) 0x24;
146     ioPORTB = (uint8_t *) 0x25;
147
148     pinPB0State = BUTTON_NOT_PRESSED;
149
150       //make PB1 as output
151     *ioDDRB = 0x02; // DDRB[7:0] 0000 0010
152
153     //Enable internal pull up for PB0
154     *ioPORTB = 0x01;
155
156
157   }
158
159   void turnOnPB1LED() {
160     *ioPORTB = (*ioPORTB) | 0x02; // 0000 0010
161   }
162
163   void turnOffPB1LED() {
164     *ioPORTB = (*ioPORTB) & 0b11111101; // 0000 0000
165   }
166
167   uint8_t readInputPB0() {
168     return ((*ioPINB) & 0x01) >> 0;
169   }
170
171   int debouncePB0() {
172     uint8_t currentPB0Val = readInputPB0();
173
174     if (currentPB0Val != pinPB0State) {
175       // have a potential pin change!!
176       myHardDelay(50); // wait for bounce to end
177       currentPB0Val = readInputPB0();
178       if (currentPB0Val != pinPB0State) { // if still diff, then it was a transtient and
              was an actual pin changed
179         pinPB0State = currentPB0Val;
180         return 1; // pin changed
181       }
182     }
183     return 0; // pin didn't changed
184   }
185
186   /****************Button**********************/
187
188   void flushRxUartBuffer() {
189     memset(rxUartBuffer, 0, sizeof(rxUartBuffer));
190     uartState = READING;
191     rxUartIndex = 0;
192   }
193
194   void mySerialBegin(uint32_t baudrate) {
195     pUCSRnA = (uint8_t *) 0xC0;
```

```
196   pUCSR0B = (uint8_t *) 0xC1;  // USART Control and Status Register 0 B
197   pUCSR0C = (uint8_t *) 0xC2;  // USART Control and Status Register 0 C
198   pUBRR0L = (uint8_t *) 0xC4;
199   pUBRR0H = (uint8_t *) 0xC5;
200   pUDRn   = (uint8_t *) 0xC6;
201
202   #define TXEN0 3
203   #define RXEN0 4
204   #define RXCIE0 7 // RX complete interupt enable bit
205
206   #define UCSZ0_01 1
207
208   uint32_t ubrr = BAUD2UBRR(baudrate);
209   *pUBRR0H = (uint8_t) (ubrr >> 8);
210   *pUBRR0L = (uint8_t) ubrr;
211
212   *pUCSRnA = 0x00;
213
214   // b[1]Enable receiver and b[0]transmitter
215   *pUCSR0B = (1<<RXEN0) | (1<<TXEN0);
216
217   // Set frame format: 8bit data, default 1stop bit
218   *pUCSR0C = (3 << UCSZ0_01);
219 }
220
221 void mySerialWriteOne(uint8_t data) {
222   #define UDREn 5 // USART Data Register Empty
223
224   /* Wait for empty transmit buffer */
225   while ( !( (*pUCSRnA) & (1<<UDREn)) );
226
227   /* Put data into buffer, sends the data */
228   *pUDRn = data;
229 }
230
231
232
233 boolean serialAvailable() {
234   #define RXC 7 // USART Receive complete,
235   return ((*pUCSRnA) & (1 << RXC)) == (1 << RXC);
236 }
237
238
239 uint8_t serialRead() {
240   return *pUDRn;
241 }
242
243 void mySerialWrite(uint8_t * msg) {
244   while ((*msg) != 0) {
245     mySerialWriteOne(*msg);
246     msg++;
247   }
248 }
249
250
251 MathQuestion generateRandomMathQuestion() {
252   MathQuestion question;
253   question.mathOperator = (rand() % 2) == 0 ? '*' : '/';
254   int num1 = (rand() % 10) + 1, num2 = (rand() % 10) + 2;
255   if (question.mathOperator == '*') {
256     question.num1 = num1;
257     question.num2 = num2;
258     question.expectedResult = num1 * num2;
259   } else {
260     question.num1 = num1 * num2;
261     question.num2 = num1;
262     question.expectedResult = num2;
263   }
```

```
264   return question;
265 }
266
267
268 void myHardDelay(uint32_t ms) {
269   volatile int16_t count;
270
271   while (ms) {
272     for (count = 0; count < 835; count++);
273     ms -= 1;
274   }
275 }
```