

# Párhuzamos és eseményvezérelt programozás

## Házi Feladat

Berta Máté, Szűcs Viktor

# Tartalom

Feladat .....	3
Kiírás .....	3
Értelmezés .....	3
Megoldás .....	4
FreeRTOS.....	4
LCDTask .....	4
gameTask .....	4
gameOverTask .....	5
Linux .....	6
gameTask .....	6
difficultyTask .....	7
mapgenTask .....	7
Program futtatása.....	7
Mellékletek.....	8
FreeRTOS.....	8
main.c .....	8
lcdDraw.h .....	10
lcdDraw.c .....	10
uart.h.....	11
uart.c .....	11
Linux .....	14
main.c .....	14
mapgen.h .....	15
mapgen.c .....	15
highscores.h.....	16
highscores.c .....	16
game.h .....	19
game.c .....	19
difficulty.h .....	22
difficulty.c .....	22
command.h .....	23
command.c .....	23

# Feladat

## Kiírás

A feladat egy banán kosár nevű játék implementációja STK3700-as fejlesztőkártya és PC felhasználásával. A fejlesztőkártya szoftverének FreeRTOS operációs rendszeren kell futnia, a PC-n futó szoftver Linux operációs rendszeren kell futnia.

A játék során az STK3700-as fejlesztőkártya LCD kijelzőjén megjelenő „banánok” elkapása a feladat. A kosarat az LCD kijelző alsó szegmensének vízszintes vonala jelképezi. A játékos 10 étellel kezdi a játékot és legfeljebb 99 banánt kaphat el. A kijelzőn ennek látszania kell.

A játék indítása és a kosár mozgatása a PC-ről UART-on érkező karakterekkel történik. ('b'-balra, 'j'-jobbra, 's'-indítás).

A játék során a banánok egyre gyorsabban jelennek meg, a sebesség gyorsulását a játék nehézségi szintje határozza meg. Három ilyen szint van.

A játékról statisztikát vezet a PC-s program, a különböző szintek legjobb eredményei és az átlagos elért pontszám van nyilván tartva.

## Értelmezés

A feladat az előző félévben elkészített beágyazott és ambiens rendszerek házi feladatának kiegészített változata. Amennyiben van rá lehetőség az előző félévben elkészített szoftvert fel kell használni.

A feladat jól felbontható részfeladatokra. STK3700-as kártyán az LCD vezérlése, UART kezelése, játék mechanizmus számítása és a játék végének kezelése. PC-n a statisztika beolvasása/kiírása, banán megjelenési helyek generálása (minden pálya egyedi) nehézségi fok szerinti sebesség növelés kiadása, UART és STDIN összekapcsolása.

# Megoldás

## FreeRTOS

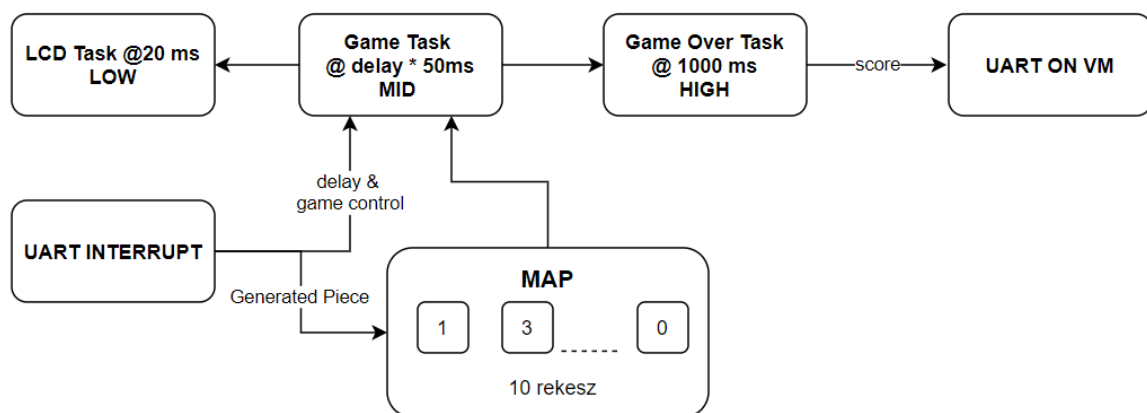
A játék megvalósítása egy interrupt kezelő függvénnyel és három taskkal történik. Az interrupt kezelő függvény fogadja és kezeli az UARTon érkező karaktereket. A taskok végzik a játék logikájának implementációját, LCD kijelző vezérlését és a játék újraindításához szükséges teendőket.

A **main()** függvény az UART és LCD inicializálása után, létrehozza a taskokat és elindítja a schedulert. Több feladattal nem rendelkezik.

A soros porton érkező karakter globális változók értékét állítja a következő táblázat szerint:

Karakter	Változó
's'	gameIsRunning = 1
'a'	position--
'd'	position++
'+'	delayTimeFactor--
'_'	delayTimeFactor++
'0','1','2','3'	map[mapIndex] = 'c'-0x30

A beágyazott szoftver folyamatábrája:



## LCDDTask

Legalacsonyabb prioritással rendelkező, viszont leggyakrabban futó task. Feladata, ha bármilyen változás történik a képernyőkép újra rajzolása az LCD-n. Ezt 20ms-os frissítéssel teszi meg, amennyiben nincs más magasabb prioritású task.

## gameTask

Közepes prioritású task, amely a delayTimeFactortól függően 50ms és 500 ms ideig várakozás két futási ciklus között.

A játék logikáját valósítja meg, mint például a banán és kosár pozíciójának összehasonlítása, banán érése, új banán létrehozása.

### gameOverTask

Legmagasabb prioritású task, amely másodpercenként fut. Amennyiben a játékosnak elfogytak az életpontjai vagy elérte a 99-es maximális pontszámot, UART-on a PC felé továbbítja az eddig elért pontszámot. A továbbítás után a játékot vezérlő változók alaphelyzetbe állítása a feladata.

## Linux

A program 6 c file-t és a hozzájuk tartozó 5 header file-t tartalmaz és 3 taskot használ.

A **main.c** file az UART inicializálása után, létrehozza a taskokat. Több feladattal nem rendelkezik.

A **command.c** file a soros port felkonfigurálásához szükséges paramétereket kezeli.

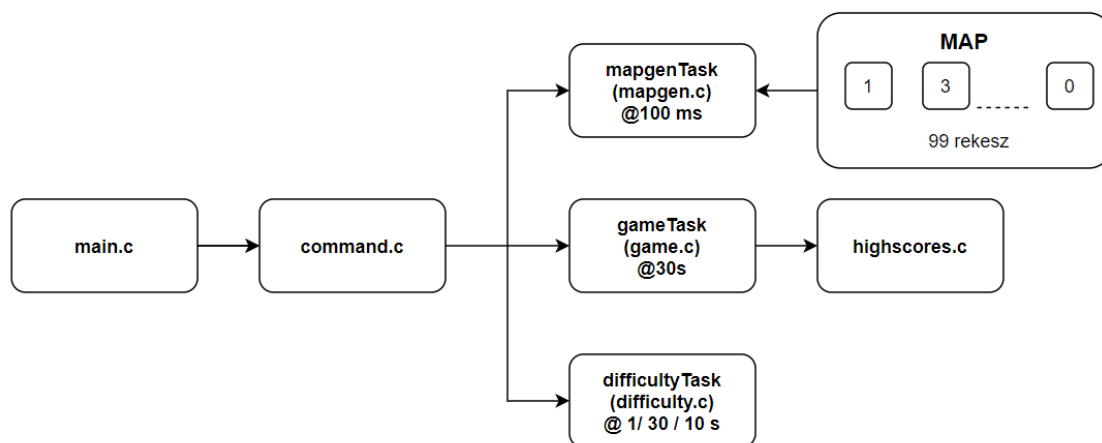
A **game.c** file tartalmazza a gameTask-ot, valamint a gameOver() függvényt, mely a highscores számítását, megjelenítését és az új játék előkészítését végzi.

A **difficulty.c** file kezeli a nehézségi módok beállítását, '+' karakterek küldésével, mellyel a delayTimeFactor változót állítja a kártyán.

A **mapgen.c** file tartalmazza a mapgenTask-ot, valamint elvégzi a pálya generálásához szükséges randomszám generálást

A **highscores.c** file a játékstatisztikák fájlba írását és olvasásának feladatát végzi.

A szoftver folyamatábrája:



A futtatáshoz szükséges karakterparancsok:

Karakter	Változó
'h'	Help
's'	Soros port sebessége
'p'	Soros port neve

## gameTask

Feladata, a játék indításának vezérlése, a leütött karakterek soros portra történő kiírása.

### difficultyTask

A delayTimeFactor növeléséhez a '+' karakter kiírása a soros portra, különböző időközönként nehézségi módtól függően..

### mapgenTask

A randomszám generáló függvény által feltöltött tömb elemeit írja ki a soros portra.

### Program futtatása

A program a következő paranccsal fordítható:

```
gcc -pthread command.c difficulty.c game.c highscores.c main.c mapgen.c -o main
```

Futtatása pedig a következő paranccsal történik:

```
./main -p /dev/ttyACM0 115200
```

A soros porti eszköz neve változhat, ezért ellenőrizni kell azt.

# Mellékletek

A projekt forrásfájljai (emlib és FreeRTOS könyvtári források nélkül) a következő linken is elérhetőek:

<https://github.com/Zaion-BM/PEP>

## FreeRTOS

main.c
<pre>1 #include "em_device.h" 2 #include "em_chip.h" 3 4 #include &lt;stdint.h&gt; 5 #include "segmentlcd.h" 6 #include &lt;udelay.h&gt; 7 8 #include "FreeRTOS.h" 9 #include "task.h" 10 #include "../application/lcdDraw.h" 11 #include "../application/mapgen.h" 12 #include "../application/uart.h" 13 14 extern uint8_t position; 15 extern uint8_t delayTimeFactor; 16 extern uint8_t gameIsRunning; 17 extern uint8_t reDrawLCD; 18 19 #define  maintASK_HI_PRIORITY    ( tskIDLE_PRIORITY + 3 ) 20 #define  maintASK_MID_PRIORITY  ( tskIDLE_PRIORITY + 2 ) 21 #define  maintASK_LO_PRIORITY   ( tskIDLE_PRIORITY + 1 ) 22 #define  maintASK_LO_STACK_SIZE configMINIMAL_STACK_SIZE 23 24 #define LIFE_TO_LOSE 10 25 26 //Globális változók 27 TaskHandle_t LCDHandle; 28 TaskHandle_t gameHandle; 29 TaskHandle_t gameOverHandle; 30 31 uint8_t score = 0; //elért pontszám 32 uint8_t bananaY = 0; //banán függőleges helyzete 33 uint8_t bananaX = 0; 34 uint8_t round = 0; //körök száma 35 uint8_t life = LIFE_TO_LOSE; 36 extern uint8_t map[10]; 37 38 39 void LCDTask() { 40 41     while(1){ 42 43         if(reDrawLCD){ 44 45             lcdClearLowerSegment(); 46             lcdDrawBasket(position); 47             lcdDrawBanana(bananaX,bananaY); 48         } 49     } 50 }</pre>



```
46         lcdDrawScore(score,life);
47         reDrawLCD = 0;
48     }
49
50     vTaskDelay(20/portTICK_PERIOD_MS);
51 }
52 }
53
54 void gameTask() {
55
56     while (1) {
57         if((gameIsRunning) && (life!=0)){
58             reDrawLCD = 1;
59
60             //Banán érés
61             if(bananaY<4){bananaY++;}
62             else{bananaY=0;}
63
64             //Elkapás
65             if(4==bananaY){ (position==bananaX) ? score++ : life--; }
66
67             //Új banán rajzolás
68             if(4==bananaY){
69                 round++;
70                 bananaX = map[(round%10)];
71             }
72
73             vTaskDelay(delayTimeFactor*50/ portTICK_PERIOD_MS);
74         }
75
76         vTaskDelay(250/ portTICK_PERIOD_MS);
77     }
78 }
79
80 void gameOverTask() {
81     while(1){
82         if((life==0) || (score==99)){
83             if(score>9){
84                 USART_Tx(USART0, (score/10)+0x30);
85                 USART_Tx(USART0, (score%10)+0x30);
86             }
87             else{USART_Tx(USART0, score+0x30); }
88             score = 0;
89             round = 0;
90             delayTimeFactor = 10;
91             life = LIFE_TO_LOSE;
92             gameIsRunning = 0;
93             for(int i = 0; i<10; i++){map[i] = i%4;}
94         }
95         vTaskDelay(1000/portTICK_PERIOD_MS);
96     }
97 }
98 }
99
100
101 int main(void) {
```

```
102  /* Chip errata */
103  CHIP_Init();
104
105  SegmentLCD_Init(false);
106  uart_init();
107
108  xTaskCreate(LCDTask, "", mainTASK_LO_STACK_SIZE, NULL, mainTASK_LO_PRIORITY, &LCDHandle);
109  xTaskCreate(gameTask, "", mainTASK_LO_STACK_SIZE, NULL, mainTASK_MID_PRIORITY, &gameHandle);
110  xTaskCreate(gameOverTask, "", mainTASK_LO_STACK_SIZE, NULL, mainTASK_HI_PRIORITY, &gameOverHandle);
111
112  vTaskStartScheduler();
113
114  return 0;
115 }
```

## lcdDraw.h

```
1  #ifndef LCDDRAW_H
2  #define LCDDRAW_H
3
4
5  #include "em_device.h"
6  #include "em_chip.h"
7  #include "segmentlcd.h"
8
9  #include "../application/segmentlcd_individual.h"
10
11
12  void lcdDrawBasket(uint8_t position); //Kosár kijelzése LCD-n
13
14  void lcdDrawScore(uint8_t score, uint8_t round); //Körök száma és pontszám kijelzése LCD-n
15
16  void lcdDrawBanana(uint8_t position, uint8_t age); //Banán kijelzése LCD-n
17
18  void lcdClearLowerSegment(); //Alsó szegmens törlése
19
20
21 #endif
```

## lcdDraw.c

```
1  #include "../application/lcdDraw.h"
2
3  SegmentLCD_UpperCharSegments_TypeDef upperCharSegments[SEGMENT_LCD_NUM_OF_UPPER_CHARS];
4  SegmentLCD_LowerCharSegments_TypeDef lowerCharSegments[SEGMENT_LCD_NUM_OF_LOWER_CHARS];
5
6  void lcdClearLowerSegment()
7  {
8      for (uint8_t p = 0; p < SEGMENT_LCD_NUM_OF_LOWER_CHARS; p++)
9      {
10          lowerCharSegments[p].raw = 0;
11          SegmentLCD_LowerSegments(lowerCharSegments);
12      }
13  }
14
```

```
15 void lcdDrawBasket(uint8_t position)
16 {
17     lowerCharSegments[position].d = 1;
18     SegmentLCD_LowerSegments(lowerCharSegments);
19 }
20
21
22 void lcdDrawScore(uint8_t score, uint8_t life)
23 {
24     SegmentLCD_Number(life*100 + score);
25     SegmentLCD_Symbol(LCD_SYMBOL_COL10,1);
26 }
27
28
29 void lcdDrawBanana(uint8_t position,uint8_t age)
30 {
31     switch(age)
32     {
33     case 0:
34         lowerCharSegments[position].a = 1;
35         SegmentLCD_LowerSegments(lowerCharSegments);
36         return;
37     case 1:
38         lowerCharSegments[position].j = 1;
39         SegmentLCD_LowerSegments(lowerCharSegments);
40         return;
41     case 2:
42         lowerCharSegments[position].p = 1;
43         SegmentLCD_LowerSegments(lowerCharSegments);
44         return;
45     case 3:
46         return;
47     }
48 }
```

## uart.h

```
1 #ifndef UART_H
2 #define UART_H
3
4
5 void UART0_RX_IRQHandler(void);
6
7 void uart_init();
8
9 #endif
```

## uart.c

```
1 #include "../application/uart.h"
2
3 #include <em_usart.h>
4 #include <em_cm_u.h>
5 #include <em_gpio.h>
```

```
6
7 volatile char newchar = '0'; // Soros porton érkező karakter
8 volatile uint8_t position = 0; // Kosár pozíció
9 volatile uint8_t gameIsRunning = 0; // Játékengedélyező flag
10 volatile uint8_t delayTimeFactor = 10; // Játék sebesség szabályozó
11 volatile uint8_t newPiece = 0;
12 volatile uint8_t reDrawLCD = 1;
13 volatile uint8_t map[10] = {3,2,0,1,1,0,3,3,0};
14 volatile uint8_t mapIndex = 0;
15
16 void UART0_RX_IRQHandler(void) // UART INT HANDLER
17 {
18     newchar = (char) USART_RxDataGet(UART0);
19     if(newchar=='d' && position < 3){
20         position++;
21         reDrawLCD = 1;
22         newchar = 'q';
23     }
24
25     if(newchar=='a' && position > 0){
26         position--;
27         newchar = 'q';
28         reDrawLCD = 1;
29     }
30     if(newchar=='s'){
31         gameIsRunning = 1;
32         newchar = 'q';
33         reDrawLCD = 1;
34     }
35     if(newchar=='+'){
36         if(delayTimeFactor != 1){delayTimeFactor--;}
37         newchar = 'q';
38     }
39     if(newchar=='-'){
40         if(delayTimeFactor != 9){delayTimeFactor++;}
41         newchar = 'q';
42     }
43     if( ((newchar-0x30)<4) && ((newchar-0x30)>=0) ){
44         if(mapIndex==10){mapIndex = 0;}
45         map[mapIndex] = newchar-0x30;
46         mapIndex++;
47     }
48     else{
49         newchar = 'q';
50     }
51 }
52
53 void uart_init()
54 {
55 //UART0 konfigurálása a 4 gyakorlat kódja alapján:
56 //Configure UATR0: 11520 Baud, Frame format 81N
57 //Location 1 routing
58
59 //Enable CLK for UATR0
60 CMU_ClockEnable(cmuClock_UART0, true);
61
```

```
62     //Actual configuration
63     USART_InitAsync_TypeDef uinit;
64
65     uinit.autoCsEnable = false;
66     uinit.baudrate = 115200;
67     uinit.databits = usartDatabits8;
68     uinit.enable = usartEnable;
69     uinit.mvdis = false;
70     uinit.oversampling = usartOVS16;
71     uinit.parity = usartNoParity;
72     uinit.prsRxCh = usartPrsRxCh0;
73     uinit.prsRxEnable = false;
74     uinit.refFreq = 0;
75     uinit.stopbits = usartStopbits1;
76
77     USART_InitAsync(UART0, &uinit);
78
79     UART0->ROUTE |= (USART_ROUTE_TXPEN | USART_ROUTE_RXPEN);
80     UART0->ROUTE |= (USART_ROUTE_LOCATION_LOC1);
81
82     CMU_ClockEnable(cmuClock_GPIO, true);
83     GPIO_PinModeSet(gpioPortE, 0, gpioModePushPull, 1); // TX
84     GPIO_PinModeSet(gpioPortE, 1, gpioModeInput, 0);    // RX
85     GPIO_PinModeSet(gpioPortF, 7, gpioModePushPull, 1); // Enable to debugger
86
87     USART_IntEnable(UART0, USART_IF_RXDATAV);
88     //Interrupt enable
89     NVIC_EnableIRQ(UART0_RX_IRQn);
90
91
92 }
```

## Linux

### main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <pthread.h>
5 #include <fcntl.h>
6 #include <stdbool.h>
7 #include <stdint.h>
8 #include <unistd.h>
9 #include <termios.h>
10
11 #include "command.h"
12 #include "game.h"
13 #include "highscores.h"
14 #include "mapgen.h"
15 #include "difficulty.h"
16
17 #define CMDLINE_DBG true
18
19 #define CFGSTR_SIZE 64
20
21 // Global variables
22 int gameIsRunning = 0;
23 int ser_fd;
24
25 int main(int argc, char *argv[]) {
26
27     uint32_t baudrate = 0;
28     char      portname [CFGSTR_SIZE+1] = "";
29
30     command_interpreter(argc, argv, &baudrate, portname); //call for cmd line argument interpreter
31
32     /*Initialize serial port and open it*/
33     struct termios gecko_ser;
34
35     memset(&gecko_ser,0,sizeof(gecko_ser));
36
37     gecko_ser.c_iflag = 0;
38     gecko_ser.c_oflag = 0;
39     gecko_ser.c_cflag = CS8|CREAD|CLOCAL;
40     gecko_ser.c_lflag = 0;
41     gecko_ser.c_cc[VMIN] = 1;
42     gecko_ser.c_cc[VTIME] = 0;
43
44     ser_fd = open(portname,O_RDWR);
45
46     if(ser_fd < 0 ) {
47         perror("Serial device open failed!");
48         exit(EXIT_FAILURE);
49     }
50
51     cfsetospeed(&gecko_ser, baudrate);
52     cfsetispeed(&gecko_ser, baudrate);
53 }
```

```
54         tcsetattr (ser_fd, TCSANOW, &gecko_ser);
55
56     int status; //Return value for thread creation
57
58     /*Thread ID variables*/
59     pthread_t gameTask_ID;
60     pthread_t mapgenTask_ID;
61     pthread_t difficultyTask_ID;
62
63     /*Create threads*/
64
65     status = pthread_create(&gameTask_ID, NULL, (void*)gameTask, NULL);
66
67     status = pthread_create(&mapgenTask_ID, NULL, (void*)mapgenTask, NULL);
68
69     status = pthread_create(&difficultyTask_ID, NULL, (void*)difficultyTask, NULL);
70
71     /*Terminate only if all threads have already terminated*/
72     pthread_join(gameTask_ID, NULL);
73
74     pthread_join(mapgenTask_ID, NULL);
75
76     pthread_join(difficultyTask_ID, NULL);
77
78     exit(EXIT_SUCCESS);
79 }
```

### mapgen.h

```
1 #ifndef MAPGEN_H
2 #define MAPGEN_H
3
4 void init_map();
5
6 void mapgenTask();
7
8 #endif
```

### mapgen.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <unistd.h>
5
6 #include "mapgen.h"
7
8 int map[99]; //Memory space for map
9 extern int ser_fd; //Externally initiated UART handle
10
11 void init_map(){
12
13     srand(time(0)); //seed random time generator
14
15     for(int i = 0; i<100; i++){ //fill map array
```

```
16         map[i] = ( rand() % 4 );    //4 in order to get 0||1||2||3 numbers only
17     }
18 }
19
20 void mapgenTask() {
21
22     init_map(); //Create unique map
23     int index = 0;
24     char ch;
25
26     while(1) {
27         if(index == 100) {index=0;} //Circular buffer
28         ch = map[index]+0x30; //int to char conversion
29         write(serial_fd,&ch,1); //Send 1 byte on UART
30         index++; //Increment index
31         usleep(100000); //Sleep for 100ms
32     }
33 }
```

### highscores.h

```
1 #ifndef HIGHSCORES_H
2 #define HIGHSCORES_H
3
4 #include <stdio.h>
5 #include <string.h>
6 #include <stdlib.h>
7
8 void parse_file(FILE* fp);
9
10 void write_new_result();
11
12 void open_file();
13
14 #endif
```

### highscores.c

```
1 #include "highscores.h"
2
3 //Global constants
4 FILE* fp;
5 char temp[16];
6 char data_low[16]; // "low,00.00,00"
7 char data_mid[16]; // "mid,00.00,00"
8 char data_high[16]; // "hig,00.00,00"
9 char toplist[10][16]; // "lvl,00,name"
10
11 double avg_low, avg_mid, avg_high; // 0 at first attempt
12 int high_score_low, high_score_mid, high_score_high; // 0 at first attempt
13
14 void parse_file(FILE* fp) {
15     int i = 0;
16     while(fgets(temp, 16, fp)) {
17         if(0==i) {strcpy(data_low,temp);}
```



```
18         if(1==i) {strcpy(data_mid,temp);}
19         if(2==i) {strcpy(data_high,temp);}
20         if( (i>2) && (i<10) ) {strcpy(toplist[i],temp);}
21         i++;
22     }
23
24     //Get avg_low
25     memset(temp, '\0', sizeof(temp));
26     strncpy(temp,data_low+4,5);
27     avg_low = atof(temp);
28
29     //Get avg_mid
30     memset(temp, '\0', sizeof(temp));
31     strncpy(temp,data_mid+4,5);
32     avg_mid = atof(temp);
33
34     //Get avg_high
35     memset(temp, '\0', sizeof(temp));
36     strncpy(temp,data_high+4,5);
37     avg_high = atof(temp);
38
39     //Get hs_low
40     strncpy(temp,data_low+10,2);
41     high_score_low = atoi(temp);
42
43     //Get hs_mid
44     strncpy(temp,data_mid+10,2);
45     high_score_mid = atoi(temp);
46
47     //Get hs_high
48     strncpy(temp,data_high+10,2);
49     high_score_high = atoi(temp);
50 }
51
52 void write_new_result() {
53     fp = fopen("datafile.txt", "w"); //Open database
54     if (fp == NULL) { //Handle error
55         printf("Please call open_file() API function first.\n");
56         fclose(fp);
57     }
58     else{
59         //Write new data_low to file
60         strcpy(data_low,"low,");
61         sprintf(temp,"%05.02f",avg_low);
62         strcat(data_low,temp);
63         sprintf(temp,",%02d", high_score_low);
64         strcat(data_low,temp);
65         strcat(data_low,"\n");
66         fputs(data_low,fp);
67
68         //Write new data_mid to file
69         strcpy(data_mid,"mid,");
70         sprintf(temp,"%05.02f",avg_mid);
71         strcat(data_mid,temp);
72         sprintf(temp,",%02d", high_score_mid);
73         strcat(data_mid,temp);
```

```
74     strcat(data_mid, "\n");
75     fputs(data_mid, fp);
76
77     //Write new data_low to file
78     strcpy(data_high, "hig,");
79     sprintf(temp, "%05.02f", avg_high);
80     strcat(data_high, temp);
81     sprintf(temp, ", %02d", high_score_high);
82     strcat(data_high, temp);
83     strcat(data_high, "\n");
84     fputs(data_high, fp);
85
86     for(int i = 0; i<10;i++){
87         strcpy(toplist[i], "lvl,00,name\n"); //Create toplist element in memory
88         fputs(toplist[i], fp); //Write toplist element to file
89     }
90     fclose(fp);
91 }
92
93 }
94
95 void open_file(){
96     fp = fopen("datafile.txt", "r");
97     if (fp != NULL){
98         parse_file(fp);
99         fclose(fp);
100     }
101     else{
102         fp = fopen("datafile.txt", "w"); //Open empty file
103
104         strcpy(data_low, "low,00.00,00\n"); //Create data_low in memory
105         fputs(data_low, fp); //Write data_low to file
106
107         strcpy(data_mid, "mid,00.00,00\n"); //Create data_mid in memory
108         fputs(data_mid, fp); //Write data_mid to file
109
110         strcpy(data_high, "high,00.00,00\n"); //Create data_high in memory
111         fputs(data_high, fp); //Write data_high to file
112
113         for(int i = 0; i<10;i++){
114             strcpy(toplist[i], "lvl,00,name\n"); //Create toplist element in memory
115             fputs(toplist[i], fp); //Write toplist element to file
116         }
117
118         /*Give initial value to variables in memory*/
119         avg_low = 0;
120         avg_mid = 0;
121         avg_high = 0;
122         high_score_low = 0;
123         high_score_mid = 0;
124         high_score_high = 0;
125
126         fclose(fp);
127     }
128
129 }
```

## game.h

```
1 #ifndef GAME_H
2 #define GAME_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <unistd.h>
8 #include <termios.h>
9 #include <stdint.h>
10 #include <fcntl.h>
11
12 void gameOver();
13
14 void gameTask();
15
16 #endif
```

## game.c

```
1 #include "game.h"
2 #include "highscores.h"
3 #include "mapgen.h"
4
5 extern int gameIsRunning; //Boolean like variable
6 int difficulty_level; // 1,2,3 or 0 if no parameter was given from STDIN
7 int score = 0; //Player score during game
8
9 extern int ser_fd; //Externally initiated UART handle
10 extern double avg_low, avg_mid, avg_high; // 0 at first attempt
11 extern int high_score_low, high_score_mid, high_score_high; // 0 at first attempt
12
13 void gameOver() {
14     open_file(); //Get all data from database
15     gameIsRunning = 0; //Stop game running in other tasks
16
17     /*According to the chosen level, set avg and high score and print result to the player*/
18     switch(difficulty_level) {
19         case(1):
20             if(high_score_low <= score) {
21                 printf("Congrats! New High Score!!!!44!\n");
22                 high_score_low = score;
23             }
24             avg_low = (avg_low+score)/2;
25
26             printf("Score: %d\nAverage Score: %f\nHigh Score: %d\n\n", score, avg_low,
27                 high_score_low);
28             write_new_result();
29             break;
30         case(2):
31             if(high_score_mid <= score) {
32                 printf("Congrats! New High Score!!!!44!\n");
```

```
33         high_score_mid = score;
34     }
35     avg_mid = (avg_mid+score)/2;
36
37     printf("Score: %d\nAverage Score: %f\nHigh Score: %d\n\n", score, avg_mid,
38         high_score_mid);
39     write_new_result();
40     break;
41
42     case(3):
43         if(high_score_high <= score) {
44             printf("Congrats! New High Score!!!!44!\n");
45             high_score_high = score;
46             avg_high = (avg_high+score)/2;
47
48             printf("Score: %d\nAverage Score: %f\nHigh Score: %d\n\n", score, avg_high,
49                 high_score_high);
50             write_new_result();
51             break;
52
53         default: write_new_result();
54     }
55
56     /*Print first message to player, the next game starts here*/
57     printf("Please select game difficulty: \n"
58         "\t\t\t Easy   - press e\n"
59         "\t\t\t Normal - press n\n"
60         "\t\t\t Hard   - press h\n");
61
62     //Set game variables to default
63     score = 0;
64     difficulty_level = 0;
65     init_map(); // get a new map
66 }
67
68 void gameTask() {
69     //Control variables
70     char c = 0;
71     struct termios input;
72
73     // Disabling canonical mode and echo
74     tcgetattr(STDIN_FILENO, &input);
75
76     input.c_lflag &= ~ICANON;
77     input.c_lflag &= ~ECHO;
78
79     if(tcsetattr(STDIN_FILENO, TCSANOW, &input) == -1) {
80         perror("ERROR");
81     }
82
83     //Start the game and print first message
84     printf("Please select game difficulty: \n"
85         "\t\t\t Easy   - press e\n"
86         "\t\t\t Normal - press n\n"
87         "\t\t\t Hard   - press h\n");
```

```
87
88     while(1) {
89         /*Setting up select function to get user input*/
90         int selrval;
91         fd_set rfd;
92         struct timeval tv;
93
94         FD_ZERO(&rfd);
95         FD_SET(STDIN_FILENO, &rfd);
96         FD_SET(server_fd, &rfd);
97
98         tv.tv_sec = 30;
99         tv.tv_usec = 0;
100         selrval = select(server_fd+1, &rfd, NULL, NULL, &tv);
101
102         if(selrval < 0) {
103             perror("Select failed");
104         }
105
106         else if(selrval == 0) {
107             //printf("Select timeout...\n");
108         }
109
110         else {
111             if(FD_ISSET(STDIN_FILENO, &rfd)) { //Handle if a character is read from STDIN
112                 read(STDIN_FILENO, &c, 1);
113                 switch (c) {
114                     case ('e'):
115                         difficulty_level = 1;
116                         printf("Difficulty level Easy selected.\nPress 's' to start.\n");
117                         break;
118
119                     case ('n'):
120                         difficulty_level = 2;
121                         printf("Difficulty level Normal selected.\nPress 's' to start.\n");
122                         break;
123
124                     case ('h'):
125                         difficulty_level = 3;
126                         printf("Difficulty level Hard selected.\nPress 's' to start.\n");
127                         break;
128
129                     case ('s'):
130                         write(server_fd, &c, 1);
131                         gameIsRunning = 1;
132                         break;
133
134                     default:
135                         write(server_fd, &c, 1);
136                         break;
137                 }
138             }
139
140
141             if(FD_ISSET(server_fd, &rfd)) { //Handle if a character is read from UART
142                 read(server_fd, &c, 1);
```

```
143
144         if(score == 0) { //set score
145             score = atoi(&c);
146         }
147         else {
148             score *= 10;
149             score += atoi(&c);
150             gameOver(); //Game is over here
151         }
152     }
153 }
154 }
155
156     close(ser_fd); //Close serialport
157
158     // Re-enabling canonical mode and echo
159     input.c_lflag |= ICANON;
160     input.c_lflag |= ECHO;
161
162     if(tcsetattr(STDIN_FILENO, TCSANOW, &input) == -1) {
163         perror("ERROR");
164     }
165
166     exit(EXIT_SUCCESS);
167 }
```

## difficulty.h

```
1 #ifndef DIFFICULTY_H
2 #define DIFFICULTY_H
3
4 #include <unistd.h>
5
6 void difficultyTask();
7
8 #endif
```

## difficulty.c

```
1 #include "difficulty.h"
2
3 extern int gameIsRunning;
4 extern int difficulty_level;
5 extern int ser_fd;
6
7 void difficultyTask() {
8
9     /*Send '+' on UART periodically according to difficulty*/
10     while(1) {
11         if(gameIsRunning) {
12             switch(difficulty_level) {
13                 case(1):
14                     sleep(1);
15                     break;
16             }
```

```
17         case(2):
18             write(ser_fd, "+", 1);
19             sleep(30);
20             break;
21
22         case(3):
23             write(ser_fd, "+", 1);
24             sleep(10);
25             break;
26
27         default:
28             sleep(1);
29             break;
30     }
31 }
32 else {sleep(1);}
33 }
34 }
```

### command.h

```
1 #ifndef COMMAND_H
2 #define COMMAND_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <unistd.h>
8 #include <termios.h>
9 #include <stdint.h>
10
11 struct BNUM_speed {
12     uint32_t speed;
13     uint32_t bnum;
14 };
15
16 void command_interpreter(int argc, char*const* argv, uint32_t* baudrate, char
    *portname);
17
18 int64_t CheckSpeed(unsigned speed);
19
20 void print_help(void);
21
22 #endif
```

### command.c

```
1 #include "command.h"
2
3 #define CFGSTR_SIZE 64
4
5 struct BNUM_speed g_speed[] = {
6     { 50 , B50 },
7     { 75 , B75 },
```

```
8      { 110 , B110 },
9      { 134 , B134 },
10     { 150 , B150 },
11     { 200 , B200 },
12     { 300 , B300 },
13     { 600 , B600 },
14     { 1200 , B1200 },
15     { 1800 , B1800 },
16     { 2400 , B2400 },
17     { 4800 , B4800 },
18     { 9600 , B9600 },
19     { 19200 , B19200 },
20     { 38400 , B38400 },
21     { 57600 , B57600 },
22     { 115200 , B115200 },
23     { 230400 , B230400 },
24     { 460800 , B460800 },
25     { 500000 , B500000 },
26     { 576000 , B576000 },
27     { 921600 , B921600 },
28     { 1000000 , B1000000 },
29     { 1152000 , B1152000 },
30     { 1500000 , B1500000 },
31     { 2000000 , B2000000 },
32     { 2500000 , B2500000 },
33     { 3000000 , B3000000 },
34     { 3500000 , B3500000 },
35     { 4000000 , B4000000 },
36     { 0 , 0 } // Null termination, keep as the last element...
37 };
38
39 void command_interpreter(int argc, char*const* argv, uint32_t* baudrate, char *portname) {
40
41     int opt = 0; //cmd line variable
42     int speed; //UART speed
43
44     while((opt = getopt(argc, argv, "hs:p:")) != -1) { //Get cmd line argument and handle it
45         switch(opt) {
46             case 'h': //print help and exit
47                 print_help();
48                 exit(EXIT_SUCCESS);
49                 break;
50
51             case 's': //configure baudrate
52                 speed = CheckSpeed(atoi(optarg));
53                 if(speed == 0) {
54                     printf("ERROR: Specified serial speed is not
55 supported by termios! \n");
56                     exit(EXIT_FAILURE);
57                 }
58                 else {
59                     printf("Serial port speed: %d\n",
60 atoi(optarg));
61                     *baudrate = speed;
```



```
62             break;
63
64             case 'p': //configure port name
65                 if((int)strlen(optarg) > CFGSTR_SIZE) {
66                     printf("port: %s, long: %d\n", optarg,
67                         (int)strlen(optarg));
68                     printf("ERROR: Specified serial port name is too long!\n");
69                     exit(EXIT_FAILURE);
70                 }
71                 else {
72                     strcpy(portname, optarg);
73                     printf("Serial port name: %s\n", portname);
74                 }
75                 break;
76             }
77 }
78
79
80 int64_t CheckSpeed(unsigned speed) { //Check cmd line argument for valid UART speed
81     int i;
82
83     for (i = 0; g_speed[i].speed != speed; i++) {
84         if (g_speed[i].speed == 0) {
85             return 0;
86         }
87     }
88
89     return g_speed[i].bnum;
90 }
91
92 void print_help(void) { //Help function
93
94     printf("Banana application help: \n");
95     printf("-h: Printig application help. \n");
96     printf("-s <baudrate>: Setting serial port baudrate (speed). \n");
97     printf("-p <port name>: Setting serial port port name. \n\n");
98     printf("Game starts by pressing 's'.\n");
99     printf("To navigate the basket use the 'd' for right direction and 'a' for left direction.\n");
100
101 }
```