Dado que el máximo valor procesable por una variable de tipo carácter son 8 bytes, en decimal 128, se ha tenido que mostrar el tiempo total de juego y el de cálculo de la IA mediante una función incrustada en el código que recorre el número, mostrándolo carácter a carácter por pantalla, actualizando la posición donde debe y aislando el carácter sucesivamente del número original para poder visualizarlo donde corresponda.

Se ha empleado una función ITOA que transforma un número a ASCII, para poder así mostrarlo correctamente.

Estas son ambas funciones, la sección incrustada y el ITOA.

|  |
| --- |
| While (time\_aux > 0) { |
|  | time\_aux /= 10; |
|  | cont++; |
|  | } |
|  | if( cont < 1) cont= 1; |
|  | time\_aux=tiempo\_total; |
|  | while(cont!=0){ |
|  | int pow = cont-1; |
|  | int potencia=1; |
|  | while (pow != 0){ |
|  | potencia=potencia\*10; |
|  | pow--; |
|  | } |
|  | int number = time\_aux/potencia; |
|  | cont--; |
|  |  |
|  | Lcd\_DspAscII8x16(coor\_x+(i\*CHAR\_HOR), coor\_y + (CHAR\_VER) \* 2, BLACK, itoa(total\_ch,10,number,10)); |
|  | i++; |
|  |  |
|  | time\_aux%=potencia; |
|  | } |

|  |
| --- |
| unsigned char \*itoa\_internal(unsigned char \*buffer, size\_t len, int input, int base) { |
|  | static const char digits[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"; |
|  | unsigned char \*pos = buffer; |
|  | if (input >= base) |
|  | pos = itoa\_internal(buffer, len, input/base, base); |
|  |  |
|  | if (pos-buffer < len-1) |
|  | \*pos++ = digits[input % base]; |
|  | return pos; |
|  | } |
|  |  |
|  |  |
|  | unsigned char \*itoa(unsigned char \*buffer, size\_t len, int input, int base) { |
|  | unsigned char \*pos = buffer; |
|  |  |
|  | if (base < 2 || base > 36 || len < 1) |
|  | return NULL; |
|  |  |
|  | if (input < 0) |
|  | \*pos++ = '-'; |
|  |  |
|  | pos = itoa\_internal(pos, len, input, base); |
|  | \*pos = '\0'; |
|  | return buffer; |
|  | } |