Test de Reacción

Por Pablo Manuel Sánchez González

Descripción:

Se realizará un test de reacción a varias personas. Está formado por 2 luces leds y un pulsador, además de utilizar un Arduino MEGA 2560 y algunas resistencias de pequeño valor. El programa recogerá los datos principales del usuario y su resultado además de archivarlo.

Objetivo:

El fin de este Test es recoger resultados de distintas personas de edad variada para, más tarde, realizar un estudio sobre la capacidad de reacción y como este empeora con los años.

Guía de uso:

Al iniciar el programa nos hace la siguiente pregunta: “¿Cuántos sujetos van a realizar la prueba?”, nuestra respuesta será la dimensión del vector de estructuras denominadas ‘Sujeto’. Tras esto se abrirá un menú donde lo primero que se debe hacer es tomar los datos personales de cada sujeto (nombre, apellido y edad).
Una vez registrados todos los sujetos es posible realizar los distintos test. La opción de crear un registro no está disponible hasta que se hayan realizado los test.
Al comenzar el test el led rojo se apaga y el led verde se encenderá pasados de 2,5 a 7 segundos (número completamente aleatorio decidido en el momento para que el test sea válido). Cuando se encienda el usuario deberá pulsar el botón incorporado al hardware para apagar el led y finalizar el test.
El valor del tiempo de reacción del sujeto será almacenado en su perfil y mostrado en pantalla teniendo los milisegundos (ms) por unidad.
Una vez realizados todos los test podremos crear un archivo que almacenará y mostrará los distintos sujetos con sus datos personales y su tiempo de reacción.
Para finalizar el programa deberá seleccionar la opción de “Salir”.

Componentes de hardware:

- Arduino Mega 2560
- Elegoo Breadboard
- 3 Resistencias de 220Ω
- Pulsador de 4 patas
- 2 Leds (rojo y verde)
- Cables

Código Visual:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include "SerialClass/SerialClass.h"
#define MAX_BUFFER 200
#define PAUSA_MS 200
#define N 20

typedef struct {
	char nombre[N], apellido[N];
	int edad;
	float time;
}sujeto;

void alta_sujetos(sujeto* a, int num);
void crear_fichero(sujeto* a, int num);
float verifica_sensores(Serial* Arduino, char* port);
int Enviar_y_Recibir(Serial* Arduino, const char* mensaje_enviar, char* mensaje_recibir);
float float_from_cadena(char cadena[]);
float lanzar_test(Serial* Arduino);
int Recibir(Serial* Arduino, char* mensaje_recibir);


int main(void) {
	Serial* Arduino;
	sujeto* v;
	char puerto[] = "COM3";
	int i = 0, n, edad, opcion, hay_valores = 0, posible = 0;
	char nombre[N], apellido[N], intro;
	Arduino = new Serial((char*)puerto);
	printf("¿Cuantos sujetos van a realizar la prueba?\n");
	scanf_s("%d", &n);
	v = (sujeto*)malloc(sizeof(sujeto) * n);
	if (v == NULL)
		printf("No hay memoria disponible\n");
	else {
		do
		{
			printf("\nTEST DE REACCION");
			printf("\n=======================");
			printf("\n1 - Tomar datos personales.");
			printf("\n2 - Iniciar test");
			printf("\n3 - Crear registro");
			printf("\n4 - Salir");
			printf("\nIntroduzca opcion:");
```

```
                        scanf_s("%d", &opcion);
                        scanf_s("%c", &intro);


                        if (hay_valores == 0 && opcion > 1 && opcion < 6)
                                printf("\nAntes de hacer operaciones es necesario -> 1- Tomar
datos personales\n");
                        else
                                switch (opcion)
                                {
                                case 1:
                                        hay_valores = 1;
                                        alta_sujetos(v, n);
                                        break;
                                case 2:
                                        if (i < n) {
                                                v[i].time = verifica_sensores(Arduino, puerto);
                                                if (v[i].time != 0) {
                                                        i++;
                                                        posible = 1;
                                                }
                                        }
                                        else {
                                                printf("\nNo hay memoria para mas test\n");
                                        }
                                        break;
                                case 3:
                                        if (posible == 0) {
                                                printf("\nNo es posible crear el fichero sin
realizar antes una prueba\n");
                                        }
                                        else {
                                                crear_fichero(v, n);
                                        }
                                        break;
                                case 4:
                                        break;
                                default:
                                        printf("\nOpcion inexistente\n");
                                }
                        } while (opcion != 4);
        }
        return 0;
}

void alta_sujetos(sujeto* a, int num)
{
        int i;
```

```
	char intro;
	for (i = 0; i < num; i++) {
		printf("Dime tu nombre: \n");
		fgets(a[i].nombre, N, stdin);
		printf("Dime tu apellido: \n");
		fgets(a[i].apellido, N, stdin);
		printf("Dime tu edad: \n");
		scanf_s("%d", &a[i].edad);
		scanf_s("%c", &intro);
	}
}

void crear_fichero(sujeto* a, int num)
{
	FILE* fichero;
	int i;
	errno_t e;
	e = fopen_s(&fichero, "Listado_Sujetos.txt", "wt");
	if (fichero == NULL)
		printf("No se ha podido guardar los datos\n");
	else
	{
		for (i = 0; i < num; i++)
		{
			fprintf(fichero, "%s\n", a[i].nombre);
			fprintf(fichero, "%s\n", a[i].apellido);
			fprintf(fichero, "%d\n", a[i].edad);
			fprintf(fichero, "%.1f ms\n", a[i].time);
			fprintf(fichero, "============================\n");
		}
		fclose(fichero);
		printf("Se ha creado el fichero");
	}

}

float verifica_sensores(Serial* Arduino, char* port)
{
	float tiempo = 0;
	if (Arduino->IsConnected())
	{
		tiempo = lanzar_test(Arduino);
		if (tiempo != -1)
			printf("\nTiempo: %f\n", tiempo);
	}
	else
	{
		printf("\nNo se ha podido conectar con Arduino.\n");
```

```
		printf("Revise la conexión, el puerto %s y desactive el monitor serie del IDE
de Arduino.\n",port);
	}
	return tiempo;
}

int Enviar_y_Recibir(Serial* Arduino, const char* mensaje_enviar, char* mensaje_recibir)
{
	int bytes_recibidos = 0, total = 0;
	int intentos = 0, fin_linea = 0;
	Arduino->WriteData((char*)mensaje_enviar, strlen(mensaje_enviar));
	Sleep(PAUSA_MS);
	bytes_recibidos = Arduino->ReadData(mensaje_recibir, sizeof(char) * MAX_BUFFER
- 1);
	while ((bytes_recibidos > 0 || intentos < 5) && fin_linea == 0)
	{
		if (bytes_recibidos > 0)
		{
			total += bytes_recibidos;
			if (mensaje_recibir[total - 1] == 13 || mensaje_recibir[total - 1] == 10)
				fin_linea = 1;
		}
		else
			intentos++;
		Sleep(PAUSA_MS);
		bytes_recibidos = Arduino->ReadData(mensaje_recibir + total, sizeof(char) *
MAX_BUFFER - 1);
	}
	if (total > 0)
		mensaje_recibir[total - 1] = '\0';
	return total;
}

float float_from_cadena(char cadena[]) {
	int i, estado = 0, divisor = 10;
	float numero;
	for (i = 0; i < MAX_BUFFER || cadena[i] != '\0'; i++) {
		switch (estado){
		case 0:
			if (cadena[i] >= '0' && cadena[i] <= '9')
			{
				numero = cadena[i] - '0';
				estado = 1;
			}
			break;
		case 1:
			if (cadena[i] >= '0' && cadena[i] <= '9')
				numero = numero * 10 + cadena[i] - '0';
```

```
                        else
                                if (cadena[i] == '.' || cadena[i] == ',')
                                        estado = 2;
                                else
                                        estado = 3;
                        break;
                case 2:
                        if (cadena[i] >= '0' && cadena[i] <= '9')
                        {
                                numero = numero + (float)(cadena[i] - '0') / divisor;
                                divisor *= 10;
                        }
                        else
                                estado = 3;
                        break;
                }
        }
        return numero;
}

float lanzar_test(Serial* Arduino)
{

        float tiempo;
        int bytesRecibidos;
        char mensaje_recibido[MAX_BUFFER];

        if (Arduino->IsConnected())
        {
                bytesRecibidos = Enviar_y_Recibir(Arduino, "START_TEST\n",
mensaje_recibido);
                if (bytesRecibidos > 0)
                {
                        if (bytesRecibidos <= 0)
                        {
                                printf("\nNo se ha recibido respuesta a la petición\n");
                                tiempo = -1;
                        }
                        else
                        {
                                printf("\nLa respuesta recibida tiene %d bytes.
Recibido=%s\n", bytesRecibidos, mensaje_recibido);
                                Sleep(10000);
                                bytesRecibidos = Recibir(Arduino, mensaje_recibido);
                                if (bytesRecibidos > 0)
                                        tiempo = float_from_cadena(mensaje_recibido);
                                else
                                {
```

```
                                        printf("Timeout sin recibir respuesta del test\n");
                                        tiempo = -1;
                                }
                        }
                        return tiempo;
                }else{
                printf("\nNo se ha recibido respuesta a la petición\n");
                                tiempo = -1;
                                return tiempo;
                }
        }
}

int Recibir(Serial* Arduino, char* mensaje_recibir)
{
        int bytes_recibidos = 0, total = 0;
        int intentos = 0, fin_linea = 0;
        bytes_recibidos = Arduino->ReadData(mensaje_recibir, sizeof(char) * MAX_BUFFER
- 1);

        while ((bytes_recibidos > 0 || intentos < 10) && fin_linea == 0)
        {
                if (bytes_recibidos > 0)
                {
                        total += bytes_recibidos;
                        if (mensaje_recibir[total - 1] == 13 || mensaje_recibir[total - 1] == 10)
                                fin_linea = 1;
                }
                else
                        intentos++;
                Sleep(2 * PAUSA_MS);
                bytes_recibidos = Arduino->ReadData(mensaje_recibir + total, sizeof(char) *
MAX_BUFFER - 1);
        }
        if (total > 0)
                mensaje_recibir[total - 1] = '\0';
        return total;
}
```

Código Arduino:

```
int pulsador = 6;
int LEDV = 5;
int LEDR = 4;
int estado=LOW;
long int tiempo1 =0;
long int tiempo2 =0;
float time =0;
```

```
String mensaje_entrada;
String mensaje_salida;
int test = 0;
String tiempo;
void setup ()
{
   pinMode(pulsador, INPUT);
   pinMode(LEDV, OUTPUT);
   pinMode(LEDR, OUTPUT);
   digitalWrite(LEDV, LOW);
   digitalWrite(LEDR, HIGH);
    Serial.begin(9600); // Configura velocidad puerto serie 9600 bits/sg.
    while(!Serial) { ; } // Mientras no tenga conexión se queda en bucle.
}
void loop ()
{
    if( Serial.available()> 0) // Si hay datos disponibles en el puerto serie.
    {
      String str = Serial.readStringUntil('\n');
      if (str.compareTo("START_TEST")==0){
        str="Ok";
        test = 1;
      }
      else
        str="COMANDO DESCONOCIDO";
      Serial.println(str);

      if(test == 1){
          delay(random(2500, 7000));
          digitalWrite(LEDR, LOW);
          tiempo1 = millis();
          digitalWrite(LEDV, HIGH);
          while(digitalRead(pulsador)== LOW);
          estado = digitalRead(LEDV);
          digitalWrite(LEDV, !estado);
          tiempo2 = millis();
          time = tiempo2 - tiempo1;
          mensaje_salida = (String) time;
          while(digitalRead(pulsador)==HIGH);
        tiempo = ("El tiempo es de = %d ms", time);
        Serial.println(tiempo);
        }
    }
}
```