CONTROL DE MASCOTAS
Programa basado en C++ y arduino que permite al usuario controlar a su mascota y sus necesidades.

Objetivos del trabajo

Nuestro objetivo es realizar un programa que satisfazca las necesidades de su mascota sin la obligación de la presencia física del dueño.
El programa notificará al usuario en caso de que la temeperatura de la caseta sea muy alta, encendera la luz de la caseta una vez haya oscuridad o el usuario lo solicite, medira el nivel de agua en el recipiente, y en caso de que este este vacio, permitira su llenado.
Sensores a utilizar
- Sensor Ultrasonido
Los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción.

- Sensor de temperatura.
El LM35 es un sensor de temperatura de precisión integrado. A diferencia de otros dispositivos como los termistores en los que la medición de temperatura se obtiene de la medición de su resistencia eléctrica, el LM35 es un integrado con su propio circuito de control, que proporciona una salida de voltaje proporcional a la temperatura.
La salida del LM35 es lineal con la temperatura, incrementando el valor a razón de 10mV por cada grado centígrado. El rango de medición es de -55ºC (-550mV) a 150ºC (1500 mV). Su precisión a temperatura ambiente es de 0,5ºC.
Los sensores LM35 son relativamente habituales en el mundo de los aficionados a la electrónica por su bajo precio, y su sencillez de uso.

- Medida de intensidad luminosa- Sensor de luz LDR GL55:

Fuente:
https://www.luisllamas.es/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/ El sensor LDR es un dispositivo formado por un semiconductor y su funcionamiento es tal que, al aumentar la incidencia de la luz sobre él, disminuye su resistencia. Este cambio de resistencia se debe a que, cuando recibe luz, el semiconductor que lo forma absorbe fotones y los electrones pasan a la banda de conducción y así disminuye su resistencia. Su mayor desventaja es la poca precisión y su uso limitado, ya que no puede ser usado para medir la intesidad lumínica, solo detecta los valores de "oscuridad" y "luminosidad".
- Micro Servo 9g SG90 de Tower Pro

Dinámica de la aplicación
MENU:
1. Mostrar nivel del agua del bebedero
2. Control temperatura de la caseta
3. Iluminación
4. Control de agua.

-Opción 1
El usuario podrá ver la altura a la que esté el agua en función del recipiente.

-Opción 2
Si se supera un cierto nivel de temperatura dentro de la caseta , el programa avisará al usuario.

-Opción 3
Si se supera un cierto nivel de temperatura dentro de la caseta , el programa avisará al usuario.Las luces situadas en la fachada de la caseta sen encendaran al no haber luz de noche.

-Opción 4
Automático del bebedero en función de la opinión del usuario.

CODIGO VISUAL

```
#define MAX_BUFFER 200
#define MAX_INTENTOS_READ 4
#define MS_ENTRE_INTENTOS 250
#define SI 1
#define NO 0
#define LONGCAD 20

#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <windows.h>
#include <string.h>
#include <conio.h>
#include"SerialClass/SerialClass.h"

typedef struct
{
        float temp;
        char dia[LONGCAD];

}TEMPERATURA;


// Funciones prototipo
int menu_principal(void);
void configura(void);

void Talk_with_Arduino(Serial* Arduino);
void Send_to_hw(Serial*, char*);
int Receive_from_hw(Serial* Arduino, char* BufferEntrada);
int Send_and_Receive(Serial* Arduino, const char* msg_out, int valor_out, char*
msg_in, int* valor_in);
//void monitorizar_aforo(Serial*);
void mostrar_temp(Serial*);
void luces(Serial*);
```

```
void automatico(Serial*);
void apagar_luces(Serial*);
void encender_luces(Serial*);
void mostrar_nivel_bebedero(Serial*);
void elegir_nivel_bebedero(Serial*);
void leer_fichero_temperaturas(TEMPERATURA[], int*, int);
int escribir_fichero_temperaturas(TEMPERATURA*, int );

int main(void)
{
        Serial* Arduino; char puerto[] = "COM5"; //Puerto serie al que está
conectado Arduino int opc; // Opción del menú principal seleccionada
        // Tareas de configuración y carga
        TEMPERATURA* temperaturas;
        int n = 7;
        temperaturas = (TEMPERATURA*)malloc(sizeof(TEMPERATURA) * n);
        configura();
        Arduino = new Serial((char*)puerto);  // Establece la conexión con
Arduino

        // Bucle principal de la aplicación
        do
        {
                opc = menu_principal();
                switch (opc)
                {
                case 1:
                        mostrar_nivel_bebedero(Arduino);
                        break;
                case 2:
                        mostrar_temp(Arduino);
                        escribir_fichero_temperaturas(temperaturas, n);
                        break;
                case 3:
                        luces(Arduino);
                        break;
                case 4:
                        elegir_nivel_bebedero(Arduino);
                        break;
                case 5:
                        automatico(Arduino)break;
                }
                printf("\n\n");
        } while (opc != 6);

        // Tareas de desconexión y cierre
        return 0;
}

int menu_principal(void)
{ int opcion;
do
{
        printf("1 - Mostrar el nivel de agua del bebedero\n");
```

```
        printf("2 - Leer temperatura de la caseta\n");
        printf("3 - Automatico\n");
        printf("4 - Elegir nivel del agua\n");
        printf("5 -Encendido y apagado de luces\n");
        printf("6 - Cerrar Aplicación\n");
        printf("Seleccione opción: ");
        scanf_s("%d", &opcion);
        if (opcion < 1 || opcion>6)
                printf("\nOpción inexistente.\n\n");
} while (opcion < 1 || opcion>6);
return opcion;
}

void mostrar_temp(Serial* Arduino)
{
        int bytes_recibidos;
        char mensaje_in[200];
        TEMPERATURA* temperaturas;
        int i = 0;
        char mensaje_out[] = "MOSTRAR_TEMP";
        int temp; float temp2;
        bytes_recibidos = Send_and_Receive(Arduino, "MOSTRAR_TEMP", 1,
mensaje_in, &temp);
        if (bytes_recibidos != 0)
        {
                temp2 = (float)temp / 100; printf("\nLa temperatura de la caseta
es %.2fºC", temp2);
                (temperaturas + i)->temp = temp2;
                i++;
                if(temp2>=30){

                printf("La temperatura de la caseta es muy elevada,revise el
estado de su mascota\n");
                }
                else
                {
                printf("La temperatura de la caseta esta a una temperatura
correcta\n");
                }
}

void leer_fichero_temperaturas(TEMPERATURA  temperaturas[], int* pnumero, int
longitud)
{
        FILE* fichero; // Puntero para manipular el fichero
        int num = 0; // Variable auxiliar para numero de temps leidas3
        int i, pos; // Variable bucle y posicion final cadena
        errno_t cod_error;
        char intro[2];

        cod_error = fopen_s(&fichero, "Temperaturas.txt", "rt"); // Se intenta
abrir el fichero de texto
        if (cod_error != 0)  // Si el fichero no se ha podido abrir
                *pnumero = 0;
```

```
        else  // Si el fichero ha podido abrirse
        {
                fscanf_s(fichero, "%d", &num); // Se lee la cantidad de temps
                if (num == 0) // Si esa cantidad es cero
                        *pnumero = 0; // La lista estar  vac a
                else  // Si hay registros para leer (seg n el entero le do)
                {
                        if (num > longitud) // Si no hay memoria suficiente
                        {
                                printf("Memoria insuficiente para almacenar los
datos del fichero\n");
                                *pnumero = 0;
                        }
                        else // Si hay memoria suficiente
                        {
                                fgets(intro, 2, fichero); // Saltamos el intro
que hay tras el numero (Ascii 10 y 13)
                                for (i = 0; i < num; i++)  // Se leen los
registros uno por uno
                                {
                                        fgets((temperaturas + i)->dia, LONGCAD,
fichero);
                                        pos = strlen((temperaturas + i)->dia);
                                        (temperaturas + i)->dia[pos - 1] = '\0';
                                        fscanf_s(fichero, "%d", &(temperaturas +
i)->temp);
                                        fgets(intro, 2, fichero);
                                }
                                *pnumero = num;
                        }
                }
                fclose(fichero); // Se cierra el fichero
        }
}

int escribir_fichero_temperaturas(TEMPERATURA* temperaturas, int numero)
{
        int i;
        FILE* fichero;
        errno_t err;

        err = fopen_s(&fichero, "Temperaturas.txt", "wt");
        if (err == 0) // Si el fichero se ha podido crear
        {
                fprintf(fichero, "%d\n", numero); // Se graba en el fichero el
numero de usuarios
                for (i = 0; i < numero; i++)
                {
                        fprintf(fichero, "%s\n", (temperaturas + i)->dia);
                        fprintf(fichero, "%f\n", (temperaturas + i)->temp);
                }
                fclose(fichero);
        }
        else
```

```
                printf("Se ha producido un problema a la hora de grabar el
fichero de usuarios\n");
        return err;
}

void luces(Serial* Arduino)
{
        int bytes;
        char mensaje_in[200];
        char mensaje_out[] = "ILUMINACION";
        int luz;
        bytes = Send_and_Receive(Arduino, "ILUMINACION", 1, mensaje_in, &luz);
        if (bytes != 0)
        {
                if (mensaje_in[0] == '1') {
                        int siono;
                        printf("\nLas luces de la caseta están encendidas, desea
apagarlas?\n\t1.Si\n\t2.No\n");
                        scanf_s("%d", &siono);
                        switch (siono)
                        {
                        case 1:
                                apagar_luces(Arduino);
                                break;
                        case 2:
                                break;
                        }
                }
                else
                {
                        int siono;
                        printf("\nLas luces de la caseta están apagadas, desea
encenderlas?\n\t1.Si\n\t2.No\n");
                        scanf_s("%d", &siono);
                        switch (siono)
                        {
                        case 1:
                                encender_luces(Arduino);
                                break;
                        case 2:
                                break;
                        }
                }
        }
}

void apagar_luces(Serial* Arduino)
{
        int bytes;
        char mensaje_entr[200];
        char mensaje_sali[] = "APAGA";
        int valor;
        bytes = Send_and_Receive(Arduino, "APAGA", 1, mensaje_entr, &valor);
}
```

```
void encender_luces(Serial* Arduino)
{
        int bytes;
        char mensaje_entr[200];
        char mensaje_sali[] = "ENCIENDE";
        int valor;
        bytes = Send_and_Receive(Arduino, "ENCIENDE", 1, mensaje_entr, &valor);
}

void mostrar_nivel_bebedero(Serial* Arduino)
{
        int bytes_recibidos;
        char mensaje_in[255];
        char mensaje_out[] = "MOSTRAR_NIVEL_BEBEDERO";
        float altura = 13;
        int distancia;
        float nivel2;
        bytes_recibidos = Send_and_Receive(Arduino, "MOSTRAR_NIVEL_BEBEDERO",
-1, mensaje_in, &distancia);
        if (bytes_recibidos != 0)
        {
                nivel2 = (float)distancia / 100 ;
                nivel2 = altura - nivel2;
                printf("\nEl nivel del agua de su bebedero es %.2f cm", nivel2);

        }
}
void automatico(Serial* Arduino)
{
        elegir_nivel_bebedero(Arduino);
        int bytes_recibidos;
        char mensaje_entr[200];
        char mensaje_sal[] = "AUTOMATICO";
        int var;
        char tecla;
        bytes_recibidos = Send_and_Receive(Arduino, "AUTOMATICO", 1,
mensaje_entr, &var);
        while (!_kbhit())
        {
                automatico(Arduino);
        }
        tecla = _getch();
}

void elegir_nivel_bebedero(Serial* Arduino)
{
        int bytes_recibidos;
        char mensaje_in[255];
        char mensaje_out[] = "NIVEL_BEBEDERO_CONSIGNA";
        float nivel, x;
        int altura = 15;
        printf("\nç);
        printf("\nElija la altura mínima a la que quiere que este el agua de su
bebedero (en cm) :\n");
```

```
        scanf_s("%d", &x);
        nivel = altura - x;
        bytes_recibidos = Send_and_Receive(Arduino, "MOSTRAR_NIVEL_AGUA", nivel,
mensaje_in, &altura);

}

void configura(void)
{
        // Establece juego de caracteres castellano
        // Para que funcione hay que partir de un proyecto vacío
        // No utilice la plantilla Aplicación de consola C++ setlocale(LC_ALL,
"spanish");
}

// Ejemplo de función de intercambio de datos con Arduino
void Talk_with_Arduino(Serial* Arduino)
{ //char BufferSalida[MAX_BUFFER];
        char BufferEntrada[MAX_BUFFER];
        int bytesReceive,numero_recibido;
        if (Arduino->IsConnected()) // Si hay conexión con Arduino
        {

                // Para enviar un mensaje y obtener una respuesta se utiliza la
función Send_and_Receive
                // El mensaje está formado por un texto y un entero
                // El mensaje que se recibe está formado también por un texto y
un entero.
                // Parámetros de la función:
                // El primero es la referencia a Arduino
                // El segundo es el mensaje que se desea enviar
                // El tercero es un entero que complementa al mensaje que se
desea enviar
                // El cuarto es el vector de char donde se recibe la respuesta
                // El quinto es la referencia donde se recibe el entero de la
respuesta
                // La función devuelve un entero con los bytes recibidos. Si es
cero no se ha recibido nada.

                bytesReceive = Send_and_Receive(Arduino, "GET_AFORO_MAX", -1,
BufferEntrada, &numero_recibido);
                if (bytesReceive == 0)
                        printf("No se ha recibido respuesta al mensaje
enviado\n");
                else
                        printf("Mensaje recibido %s %d\n", BufferEntrada,
numero_recibido);
        }
        else
                printf("La comunicación con la plataforma hardware no es posible
en este momento\n"); // Req 3

}
```

```
// Protocolo de intercambio mensajes entre Pc y platforma hardware
// Envío Mensaje valor
// Recibe Mensaje valor
// Retorna bytes de la respuesta (0 si no hay respuesta)
int Send_and_Receive(Serial* Arduino, const char* msg_out, int valor_out, char*
msg_in, int* valor_in)
{
	char BufferSalida[MAX_BUFFER];
	char BufferEntrada[MAX_BUFFER];
	char* ptr;
	int bytesReceive;
	sprintf_s(BufferSalida, "%s\n%d\n", msg_out, valor_out);
	Send_to_hw(Arduino, BufferSalida);
	bytesReceive = Receive_from_hw(Arduino, BufferEntrada);
	if (bytesReceive != 0)
	{
		ptr = strpbrk(BufferEntrada, " ");
		if (ptr == NULL)
			*valor_in = -1;
		else
		{
			*valor_in = atoi(ptr);
			*ptr = '\0';
		}
		strcpy_s(msg_in, MAX_BUFFER, BufferEntrada);
	}
	return bytesReceive;
}

// Envía un mensaje a la plataforma hardware
void Send_to_hw(Serial* Arduino, char* BufferSalida)
{
	Arduino->WriteData(BufferSalida, strlen(BufferSalida));
}

//Recibe (si existe) un mensaje de la plataforma hardware
//Realiza MAX_INTENTOS_READ para evitar mensajes recortados
int Receive_from_hw(Serial* Arduino, char* BufferEntrada)
{
	int bytesRecibidos, bytesTotales = 0;
	int intentos_lectura = 0;
	char cadena[MAX_BUFFER];
	BufferEntrada[0] = '\0';
	while (intentos_lectura < MAX_INTENTOS_READ)
	{
		cadena[0] = '\0';
		bytesRecibidos = Arduino->ReadData(cadena, sizeof(char) *
(MAX_BUFFER - 1));
		if (bytesRecibidos != -1)
		{
			cadena[bytesRecibidos] = '\0';
			strcat_s(BufferEntrada, MAX_BUFFER, cadena);
			bytesTotales += bytesRecibidos;
		}
```

```
                intentos_lectura++;
                Sleep(MS_ENTRE_INTENTOS);
        }
        return bytesTotales;
}
```

CODIGO DE ARDUINO

```
#include <Servo.h>
#include <OneWire.h>
#include <DallasTemperature.h>


#define pinTrigger 10
#define pinEcho 9
OneWire ourWire(2); //Comunico el programa con el pin 2 del Arduino
DallasTemperature sensor(&ourWire);

int analogInPin = A0;
int analogValue = 0;
int led = 13;

int TRIG = 10; //Variable que contiene el número del pin al cual conectemos la
señal "trigger"
int ECO = 9; //Variable que contiene el número del pin al cual conectamos la
señal "echo"
int estado = 0;
int altura = 15; //Constante altura a la que se encuentra el sensor ultrasonido
de la base de la piscina
int DURACION; //Variable duración para la función nivel_bebedero_consigna
float DISTANCIA; //Variable distancia para la función nivel _bebedero_consigna
Servo servo1 ;
float distancia; //Variable distancia para la función mostrar_nivel_bebedero, de
tipo float que contendrá la distancia en cm
int numero2;
long tiempo; //Variable tiempo para la función mostrar_nivel_bebedero


void setup()
{
  delay(1000);
  Serial.begin(9600);

  sensor.begin();
  pinMode(led, OUTPUT);

  pinMode(pinTrigger, OUTPUT); //Configuramoms el pin de "trigger" como salida
  pinMode(pinEcho, INPUT);  //Configuramoms el pin de "echo" como entrada
  digitalWrite(pinTrigger, LOW);//Ponemos en voltaje bajo(0V) el pin de
"trigger"
```

```
}

void loop()
{

  if (Serial.available() > 0)  // Si hay mensajes procedentes del PC
    procesar_mensaje();

  switch (estado) {
    case 0: break;
    case 1 :
      if (medir_distancia() > numero2) {
        servo1.write(0);
        estado = 0;
      }
  }


}
// Resto de acciones


float medir_distancia(void)
{
  float DISTANCIA;
  long DURACION;

  digitalWrite(TRIG, HIGH);
  delay(1);
  digitalWrite(TRIG, LOW);
  DURACION = pulseIn(ECO, HIGH);
  DISTANCIA = DURACION / 58.2;
  return (DISTANCIA);
}


void procesar_mensaje(void)
{
  int numero1;
  int numero2;
  int numero3;
  float temp;
  String cadena = Serial.readStringUntil('\n'); // Lee mensaje
  String valor = Serial.readStringUntil('\n');  // Lee valor
  numero1 = valor.toInt();  // Transforma valor a entero
  numero2 = valor.toInt();
  numero3 = valor.toInt();

  if (cadena.equals("MOSTRAR_TEMP")) // Entre las comillas se pone el texto del
mensaje que se espera
  {
    sensor.requestTemperatures();
    temp = sensor.getTempCByIndex(0);
    temp = temp * 100;
```

```
    numero1 = int(temp);
    Serial.println(numero1);
  }
  else if (cadena.equals("ILUMINACION")) // Y así sucesivamente con todos los
posibles mensajes
  {
    numero1 = digitalRead(led);
    Serial.println(numero1);

  }
  else if (cadena.equals("AUTOMATICO"))
  {
    analogValue = analogRead(analogInPin);
    Serial.println(analogValue);
    delay(10);
    if (analogValue < 800) {
      digitalWrite(led, HIGH);
    } else {
      digitalWrite(led, LOW);
    }

  }
  else if (cadena.equals("APAGA"))
  {
    digitalWrite(led, LOW);
  }
  else if (cadena.equals("ENCIENDE"))
  {
    digitalWrite(led, HIGH);
  }
  else if (cadena.equals("MOSTRAR_NIVEL_BEBEDERO")) // Entre las comillas se
pone el texto del mensaje que se espera
  {
    digitalWrite(pinTrigger, HIGH);//Ponemos en voltaje alto(5V) el pin de
"trigger"
    delayMicroseconds(10);//Esperamos en esta línea para conseguir un pulso de
10us
    digitalWrite(pinTrigger, LOW);//Ponemos en voltaje bajo(0V) el pin de
"trigger"

    tiempo = pulseIn(pinEcho, HIGH);//Utilizamos la función  pulseIn() para
medir el tiempo del pulso/echo
    distancia = tiempo * 0.01715; //Obtenemos la distancia considerando que la
señal recorre dos veces la distancia a medir y que la velocidad del sonido es
343m/s
    if (distancia >= 140) {
      distancia = 13;
    }
    distancia = distancia * 100;
    numero2 = int(distancia);
    Serial.println("distancia ");
    Serial.println(numero2);
    delay(10);          //Nos mantenemos en esta línea durante 100ms antes de
terminar el loop
```

```
  }
  else if (cadena.equals("NIVEL_BEBEDERO_CONSIGNA")) // Y así sucesivamente con
todos los posibles mensajes
  {
    digitalWrite(TRIG, HIGH);
    delay(1);
    digitalWrite(TRIG, LOW);
    DURACION = pulseIn(ECO, HIGH);
    DISTANCIA = DURACION / 58.2;
    DISTANCIA = DISTANCIA * 100;
    //Serial.println(DISTANCIA);
    delay(200);
    // numero= altura- numero2;
    //Serial.println(numero2);
    if (DISTANCIA <= numero2 && 0 <= DISTANCIA)
    {
      estado = 1;
      servo1.write(90);
    }
    else {
      servo1.write(0);
    }
  }
}
```