TRABAJO INFORMATICA CURSO 18/19 CRUPO A109

**AUTOR N.º Matricula EMAIL**

Jaime López Arjonilla 54701 j.larjonilla@alumnos.upm.es

**CAMARA TÉRMICA CON PROCESSING Y ARDUINO**

**RESUMEN**

Sistema electrónico basado en microcontrolador capaz de mover de forma controlada dos servomotores. Los servomotores están integrados en una estructura en la que se acopla una cámara web, para fotografiar un objeto, y un sensor de temperatura infrarrojo a distancia. El sistema debe comunicarse con el ordenador y mostrar una comparativa con la imagen del objeto y un diagrama de colores en función de la temperatura de cada punto, tomada por el sistema mediante un movimiento de escaneo/barrido.

REQUISITOS FUNCIONALES

1.- El sistema dispone de una aplicación de control por pantalla para iniciar el proceso.

2.- Al iniciar la aplicación en el ordenador, el microcontrolador comienza a escanear.

3.- En cada giro combinado de los servomotores, el sensor de temperatura toma una medición.

4.- Los valores (x,y,temperatura) se envían al ordenador, que los interpreta y los dibuja en pantalla.

**HARDWARE**

**1.TERMÓMETRO INFRARROJO – MLX90614**

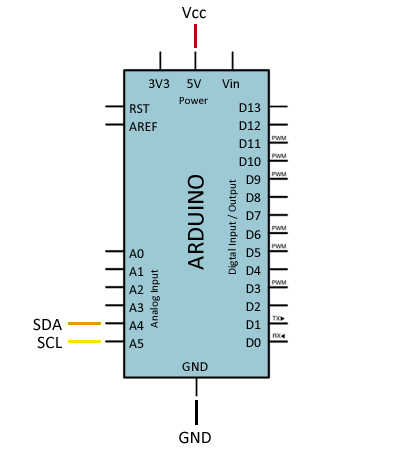
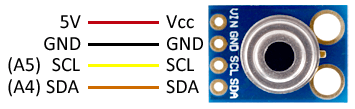
*Fuente:* <https://www.luisllamas.es/arduino-y-el-termometro-infrarrojo-a-distancia-mlx90614/>

Según la ley de Stefan-Boltzmann, todo objeto por encima del cero absoluto (K) emite radiación cuyo espectro es proporcional a su temperatura. Un sensor de temperatura a distancia es capaz de captar esta radiación y enviar una señal eléctrica proporcional a la temperatura de todos los objetos en su campo de visión. El sensor captura la temperatura de todo lo que se encuentra en su rango de visión y devuelve un valor medio del conjunto.

Los precios de los sensores de temperatura varían en función de las prestaciones de las que se quiera conseguir, es decir un sensor de temperatura con un gran ángulo de visión será más económico, pero será más útil en proyectos en los que se necesite saber con precisión si hay variaciones de temperatura, mientras que uno con un ángulo de visión más reducido dará mediciones mucho más precisas, pero con mayor coste.

El usado en este proyecto el MLX90614ESF-BAA, con una tensión de alimentación de alimentación de 3V, un único sensor infrarrojo y filtro interno. Internamente está constituido con un chip de silicio con una fina membrana micro mecanizada sensible a la radiación infrarroja, junto con la electrónica necesaria para amplificar y digitalizar la señal y calcular la temperatura. El ángulo de medición es de 80ºy mide un área de 0.83 metros de diámetro, a una distancia de 0.5 metros del lugar de medición.

Fotografía del sensor y un esquema del montaje con los pines de conexión con la plataforma Arduino.



**NOTA:** Es importante tener en cuenta la lectura del sensor solo es estable cuando el sensor se encuentra en equilibrio térmico con el ambiente, además de que necesita un tiempo de 500 ms para tomar una medición correcta.

#include <Wire.h>

#include <Adafruit\_MLX90614.h>

Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();

void setup() {

Serial.begin(9600);

mlx.begin();

}

void loop() {

Serial.print("Ambiente = ");

Serial.print(mlx.readAmbientTempC());

Serial.print("ºC\tObjeto = ");

Serial.print(mlx.readObjectTempC());

Serial.println("ºC");

delay(500);

}

Código de ejemplo de funcionamiento sensor termico

**2.MECANISMO DE ESCANEO – SERVOMOTORES**

*Fuente*: https://www.luisllamas.es/controlar-un-servo-con-arduino/

Un servomotor es un accionador ampliamente empleado en electrónica. A diferencia de otros

tipos de motores en los que se controla la velocidad de giro, en un servo se indica el ángulo

deseado y el servo se encarga de posicionarse en este ángulo.

Típicamente los servos disponen de un rango de movimiento de entre 0 a 180°. Es decir, no

son capaces de dar la vuelta por completo (de hecho, disponen de topes internos que limitan el rango de movimiento). Internamente un servo está constituido por un motor de corriente

continua, acoplado a un reductor para reducir la velocidad de giro, junto con la electrónica

necesaria para controlar su posición. Frecuentemente se dispone de un potenciómetro unido

al eje del servo que permite al servo conocer la posición del eje. Esta información es tratada

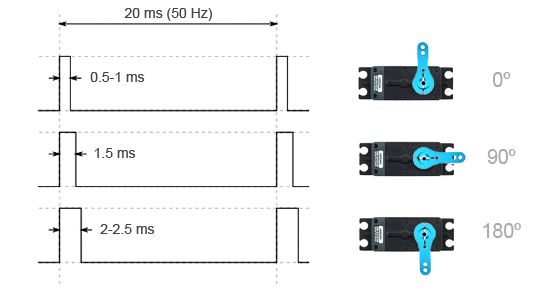
por un controlador integrado que se encarga de actuar sobre el motor para alcanzar la

posición deseada.

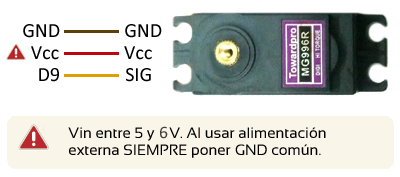
La comunicación de la posición deseada se realiza mediante la transmisión de una señal

pulsada con periodo de 20ms. El ancho del pulso determina la posición del servo. La relación

entre el ancho del pulso y el ángulo depende del modelo del motor. En general, en todos los modelos un pulso entre 500-1000μs corresponde con 0°, un pulso de 1500μs corresponde con 90° (punto neutro) y un pulso entre 2000-2500μs corresponde con 180°. Por tanto, variando la señal en microsegundos podemos disponer de una precisión teórica de 0.18°-0.36°, siempre que la mecánica del servo acompañe.



Para conectarlo a Arduino, el servo dispone de tres cables, dos de alimentación NEGRO/MARRON=GND y ROJO=Vcc, y uno AMARILLO de señal, que debe conectarse a cualquier pin digital del Arduino (el pin13 no se recomienda puesto que tiene resistencia interna, para iluminar el led que tiene a su lado). En general, la alimentación a los servos se realiza desde una fuente de tensión externa (una batería o una fuente de alimentación) a una tensión de 5V-6.5V, siendo 6V la tensión idónea. Arduino puede llegar a proporcionar corriente suficiente para encender un servo pequeño, suficiente para hacer unos cuantos proyectos de prueba, sin embargo, no dispone de corriente suficiente para actuar un servo grande ni varios servos pequeños.

Además, hacer excesiva fuerza con ellos puede exceder la capacidad de corriente de Arduino provocando su reinicio. 

#include <Servo.h>

Servo myservo;// crea el objeto servo

Servo myservo1;

int pos,pos1;

void setup() {

myservo.attach(3) ;// vincula el servo al pin digital 3

myservo1.attach(2); //vincula el servo al pin digital 3

myservo.write(0);

myservo1.write(0);

}

void loop() {

for (pos = 0; pos <= 180; pos += 1)

{

myservo1.write(pos);

for(pos1=0;pos1<=180;pos1+=1)

{

myservo.write(pos1);

delay(50);

}

delay(50);

}

}

Ejemplo de código para el movimiento de escaneo

**3.COMUNCACION SERIE: ARDUINO<->ORDENADOR**

La comunicación entre Arduino y el ordenador se realiza mediante cable por puerto USB, utilizando el puerto serie para mandar datos que quedan almacenados en el buffer de entrada esperando a ser leídos, de igual manera recibe los datos y los procesa.

**4.DESARROLLO DE LA APLICACIÓN DE CONTROL-PROCESSING**

DIBUJA CUADRADO

ALMACENA IMAGEN COMPARATIVA

MUESTRA INICIAR

MUESTRA BOTONES:

GUARDAR - REINCIAR

DIBUJA LAS PARTICIONES

MANDA SEÑAL EMPEZAR(1)

CLICK BOTON GUARDAR

BOTON

REINCIAR

COMPLETO

(X,Y) + TEMPERATURA

CLICK EN BOTON INICIAR

INICIO

MEDIR

**GUARDAR**

import processing.serial.\*; //se importa la bibilioteca para la comunicacion serie

import processing.video.\*; //se importa la bibilioteca para el uso de la webcam

// Declaro estados de la maquina

abstract class Estados

{

static final int INICIO = 0;

static final int RECIBIR = 1;

static final int GUARDAR = 2;

}

int estado = Estados.INICIO; // inicializo la maquina a INICIO

// Inicializo la camara

Capture cam;

// Creo estructura botones

class CBoton

{

int xpos, ypos, ancho, alto, fondo;

String etiqueta;

CBoton (int x, int y, int a, int h, int c, String e)

{

xpos = x;

ypos = y;

ancho = a;

alto = h;

fondo = c;

etiqueta = e;

}

void show ()

{

fill (fondo);

rect (xpos, ypos, ancho, alto, 7);

textSize(16);

fill(255);

textAlign(CENTER, CENTER);

text(etiqueta, xpos, ypos, ancho, alto);

stroke(0);

}

void unshow ()

{

stroke(255);

fill (255);

rect (xpos, ypos, ancho, alto);

}

int buttonClick(int x, int y)

{

int resultado;

if (x>=xpos && x<=xpos+ancho && y>=ypos && y<=ypos+alto)

resultado = 1;

else

resultado = 0;

return resultado;

}

}

// Inicializo los botones

CBoton BIniciar; // Botón Iniciar

CBoton BGuardar; // Botón Guardar

CBoton BReiniciar; //Boton Reiniciar

// Creo las particiones

class Particion //Mitad izquierda de la pantalla en la que estara la imagen de la camara web

{

int xpos, ypos, ancho, alto, fondo;

Particion (int x, int y, int a, int h, int c)

{

xpos = x;

ypos = y;

ancho = a;

alto = h;

fondo = c;

}

void unshow()

{

stroke(255);

fill (255);

rect (xpos, ypos, ancho, alto);

}

void show ()

{

stroke(0);

fill (fondo);

rect (xpos, ypos, ancho, alto);

noStroke();

image(cam, xpos, ypos); // se crea la imagen tomada por la webcam, se se dibuja en el punto (0,0)

}

}

class Colores //Mitad derecha de la pantalla donde estara un diagrama con colores en la que se indicara que puntos estan mas calientes que otros

{

int xpos, ypos, ancho, alto, fondo;

Table Puntos;

Colores (int x1, int y1, int a, int h, int c)

{

xpos = x1;

ypos = y1;

ancho = a;

alto = h;

fondo = c;

Puntos = new Table();

Puntos.addColumn("x");

Puntos.addColumn("y");

Puntos.addColumn("Temperatura");

}

void unshow()

{

stroke(255);

fill (255);

rect (xpos, ypos, ancho, alto);

}

void show ()

{

stroke(0);

fill (fondo);

rect (xpos, ypos, ancho, alto);

noStroke();

}

void almacena(int x, int y, float temp) //funcion en la que se almacenan los datos recibidos por arduino para su interpretacion

{

TableRow medida = Puntos.addRow();

medida.setInt("x", x);

medida.setInt("y", y);

medida.setFloat("Temperatura", temp);

dibuja(x, y, temp);

}

void dibuja(int x, int y, float temp) // se da un color al punto escaneado, de coordenadas (x,y), en funcion del valor recibido por el arduino

{ int xdibujo, ydibujo, ancho, alto;

ancho = 20;

alto = 24;

xdibujo = (x\*ancho)+640; // lo multiplico por el alto y por el ancho para adaptar la pantalla de los colores al escaneo de los servos.

ydibujo = y\*alto; // para asi dibujar cada punto donde le corresponde

if (temp >35.00) // en funcion del valor de temperatura este algoritmo nos dice de que color representarlo

{ rect(xdibujo, ydibujo, ancho, alto);

fill(128, 0, 0);//Rojo intenso

noStroke();

} else { if (temp>31.00)

{ rect(xdibujo, ydibujo, ancho, alto);

fill(255, 0, 0);//Rojo medio

noStroke();

} else { if (temp>29.50)

{ rect(xdibujo, ydibujo, ancho, alto);

fill(178, 34, 34);//Rojo

noStroke();

} else { if (temp>28.00)

{ rect(xdibujo, ydibujo, ancho, alto);

fill(255, 69, 0);//Naranja intenso

noStroke();

} else { if (temp>27.50) {

rect(xdibujo, ydibujo, ancho, alto);

fill(255, 140, 0);//naranja medio

noStroke();

} else { if (temp>27.00) {

rect(xdibujo, ydibujo, ancho, alto);

fill(255, 255, 0);//naranja

noStroke();

} else { if (temp>26.50) {

rect(xdibujo, ydibujo, ancho, alto);

fill(135, 206, 250);//Azul claro

noStroke();

} else { if (temp>26.00)

{ rect(xdibujo, ydibujo, ancho, alto);

fill(0, 191, 255);//Azul medio

noStroke();

} else {

rect(xdibujo, ydibujo, ancho, alto);

fill(0, 0, 139);//Azul intenso

noStroke();

}

}

}

}

}

}

}

}

}

}

// Inicializo las particiones

Particion foto;

Colores imgtermica;

class CSerie // Defino la clase de comunicacion serie

{

Serial puerto;

CSerie (Serial p)

{

puerto= p;

}

void Start ()

{

puerto.write(1); //cuando se llama a la funcion CanalSerie.start() esta manda un 1 a traves de puerto serie al arduino, señal de que debe empezar con el escaneo

}

int available()

{

return puerto.available(); //datos disponibles en el buffer del puerto serie

}

int leerMedida()

{

return puerto.read(); // Se lee un dato del buffer puerto serie

}

}

CSerie CanalSerie; // Comunicación serie con Arduino

Serial myPort;

// Ejecución

void setup()

{

size(1280, 580); //creo la ventana (ancho,alto)

if ( frame != null )

frame.setResizable(true);

background (255);

String portName = Serial.list()[0];

myPort = new Serial(this, portName, 9600); // se crea la comunicacion porpuerto serie con el arduino

CanalSerie = new CSerie(myPort);

BIniciar = new CBoton (600, 520, 80, 30, 127, "Iniciar"); // Crea el botón Iniciar

BGuardar = new CBoton (300, 520, 80, 30, 127, "Guardar"); // Crea el botón Guardar

BReiniciar = new CBoton (900, 520, 80, 30, 127, "Reiniciar"); // Crea el botón Reiniciar

foto = new Particion(0, 0, 640, 480, 255); //Crea la particion de la imagen webcam

imgtermica = new Colores(640, 0, 640, 480, 255); //crea la pantalla en blanco donde se va a dibujar el diagrama

String[] cameras = Capture.list(); //se enumera las camaras disponibles para ser utilizada por el ordenador

if (cameras.length == 0)

{ println("No hay camaras disponibles.");

exit();

} else

{ println("Camaras disponibles.");

for (int i = 0; i < cameras.length; i++)

{

println(cameras[i]);

}

cam = new Capture(this, cameras[13]); //elegimos la camara que queremos y creamos el objeto para que sea utilizado por el programa

cam.start(); //la camara empieza a funcionar

}

}

void draw()

{

int status, x, y, tentero, tdecimal;

float temp;

if (cam.available() == true)

{

cam.read(); //la camara hace una foto

}

switch (estado)

{

case Estados.INICIO:

BIniciar.show(); // se muestra el botón inicio

break;

case Estados.RECIBIR:

if (CanalSerie.available() > 4) // espera hasta que en el buffer haya 5 datos y los lee

{

status = CanalSerie.leerMedida();

print ("Estado "+status);

tentero = CanalSerie.leerMedida(); //procesamiento para volver a crear el valor de temperatura en formato real

tdecimal = CanalSerie.leerMedida();

temp = tentero + (float) tdecimal/100;

print ("Temperatura ", temp);

x = CanalSerie.leerMedida() ;

y = CanalSerie.leerMedida() ;

print (" (X,Y)=", x, ",", y);

if (status == 2)

estado = Estados.GUARDAR; //Cuando lee la señal de que ha terminado con el proceso pasa al siguiente estado

else

imgtermica.almacena(x, y, temp); //mientras que no se de la señal de finalizar, coge esos datos y los interpreta

}

break;

case Estados.GUARDAR:

BGuardar.show(); //se muestra el botón guardar

BReiniciar.show();//se muestra el botón guardar

break;

}

}

// Gestor eventos del ratón sensible a estados de la aplicacion

void mouseClicked()

{

switch (estado)

{

case Estados.INICIO:

if (BIniciar.buttonClick(mouseX, mouseY)== 1)

transicion\_Inicio\_Recibir();

break;

case Estados.GUARDAR:

if (BReiniciar.buttonClick(mouseX, mouseY) == 1)

transicion\_Guardar\_Inicio();

else

if (BGuardar.buttonClick(mouseX, mouseY) == 1);

transicion\_Guardar\_Guardar();

break;

}

}

// Transiciones entre estados

void transicion\_Inicio\_Recibir()

{

BIniciar.unshow();

CanalSerie.Start(); // se envia la señal a arduino para que comienze a funcionar

foto.show();

imgtermica.show();

estado = Estados.RECIBIR;

}

void transicion\_Guardar\_Inicio()

{

BGuardar.unshow();

BReiniciar.unshow();

foto.unshow();

imgtermica.unshow();

BIniciar.show();

estado = Estados.INICIO;

}

void transicion\_Guardar\_Guardar()

{

// Almacena la imagen en un fichero externo

// Se crea con la intencion de implementar en el futuro el mecanismo para guardar en el ordenador, el trabajo realizado por la camara termica

}

**5.APLICACION MICROCONTROLADOR - ARDUINO**

HASTA NO FINALIZAR ESCANEO

SEÑAL LEIDA == 1

INICIO

MEDIR

FIN

(X,Y)=ESQ.SUP.IZQUIERDA

SERIAL.WRITE 0

SERIAL.WRITE X

SERIAL.WRITE Y

SERIAL.WRITE TEMP

GIRA SERVOS

COMPLETO

ENVIA SEÑAL FINALIZACION = 2

REINICIA SERVOS

[CODIGO ARDUINO]

#include <Servo.h>

#include <Wire.h>

#include <Adafruit\_MLX90614.h>

Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();

enum Estados {INICIO, MEDIR, FIN}; // se enumeran los estados de la maquina

//se crean los objetos de clase servomotor

Servo myservo;

Servo myservo1;

int fin\_horizontal = 31;

int fin\_vertical = 19;

Estados Estado;

int pos, pos1;

void setup() {

Serial.begin(9600); // velocidad de comonucacion de datos por puerto serie

myservo.attach(3); // pin en el que por el que se comunica la placa con el servomotor

myservo1.attach(2);

myservo.write(0); //posicion inicial para hacer la foto al objeto

myservo1.write(16);

mlx.begin(); //comienza a funcionar el sensor de temperatura

Estado = INICIO;

}

void loop() {

int tentero, tdecimal;

float temperatura\_real;

switch (Estado)

{

case INICIO: if (Serial.available() > 0) // a la espara de recibir la señal del ordenador para empezar

if ( Serial.read() == 1) // comprobacion de la señal recibida sea la correcta

myservo.write(0); //desplaza a la posicion inicial para empezar a funcionar

myservo1.write(0);

Estado = MEDIR;

break;

case MEDIR:

for (pos = 0; pos <= fin\_horizontal; pos += 1) //empieza el procedimiento de escaneo de objeto y toma de medidas

{

myservo1.write(fin\_horizontal - pos);

for (pos1 = 0; pos1 <= fin\_vertical; pos1 += 1)

{

myservo.write(pos1);

delay(1000); //tiempo necesario para una buena realizacion de la medicion de temperatura

temperatura\_real = mlx.readObjectTempC(); // el valor de temperatura dado por el sensor es de tipo float, que no puede ser enviado en este formato

tentero = (int) temperatura\_real; // se procesa el valor y se manda en dos paquetes de datos por puerto serie

tdecimal = (int) (temperatura\_real \* 100) - (tentero \* 100);

Serial.write(0); //envio de señal de cotrol

Serial.write(tentero); //envio de valor temperatura

Serial.write(tdecimal);

Serial.write(pos); //se mandan las coordenadas del punto del que se ha tomado la temperatura

Serial.write(pos1);

}

delay(200);

}

Estado = FIN;

break;

case FIN:

Serial.write(2);// se envia la señal de que el proceso ha terminado

Serial.write(0);

Serial.write(0);

Serial.write(0);

myservo.write(0);

myservo1.write(0);

Estado = INICIO; // se reinicia el proceso para un nuevo escaneo

break;

}

}

[/code]