Práctica 3 : Seguidor Solar usando ESP32

Marlon Sneider Mora Cortes - 20152005034, Sara Valentina Barrero Medina -20191005170
Miguel Ángel Fuentes Ramírez-20182005007
Ingeniería Electrónica, *Universidad Distrital "Francisco José de Caldas"*
Bogotá, Colombia

***Resumen*—El diseño del seguidor solar requiere manejar conceptos como el protocolo protocolo SPI, I2C, PWM, así como requiere manejo de las interrupciones por temporizador en un microcontrolador, en este caso el ESP32, el concepto de de la activación de los sensores capacitivos es algo que puede presentar serios problemas así como la visualización de la hora en tiempo real en una pantalla TFT.**

***Palabras Clave*–**Esp32, PWM, LCD TFT, servomotor , RTC, Sensor Capacitivo, ADC, SPI, I2C

I. Introducción

En este informe se presenta la propuesta de solución al proyecto 3 de la asignatura Sistemas Embebidos 1, el cual consiste en la elaboración de un seguidor solar empleando la tarjeta de desarrollo ESP32, por medio de un real time clock y un servomotor para el posicionamiento del panel.

El dispositivo es utilizado para optimizar la cantidad de energía solar que inside sobre los paneles solares, sigue el movimiento del sol durante todo el día, en un movimiento angular de hasta 180°, utilizado normalmente en ubicaciones geográficas cercanas a la zona del Ecuador.

En este proyecto se emplea una pantalla TFT que permite la visualización de la forma de onda de la señal generada por el DAC, para el control de dicha pantalla se emplea la comunicación SPI, la cual consiste en una arquitectura de tipo maestro-esclavo. El dispositivo maestro (master) puede iniciar la comunicación con uno o varios dispositivos esclavos (slave), y enviar o recibir datos de ellos. Los dispositivos esclavos no pueden iniciar la comunicación, ni intercambiar datos entre ellos directamente.

En el bus SPI la comunicación de datos entre maestros y esclavo se realiza en dos líneas independientes, una del maestro a los esclavos, y otra de los esclavos al maestro, por tanto la comunicación es Full Duplex.El bus SPI requiere un mínimo de 3 líneas, además de una línea SS (Slave Select) para cada dispositivo esclavo conectado, para seleccionar el dispositivo con el que se va a realizar la comunicación.

Este proyecto incorpora 4 sensores táctiles capacitivos, el sistema de sensor táctil se construye sobre un sustrato que lleva los electrodos y las conexiones pertinentes bajo una superficie plana de protección. Cuando un usuario toca la superficie, la variación de la capacitancia se dispara y se genera una señal binaria para indicar si el toque realizado es válido. Esta variación es leída y convertida por el ADC, en el ESP32 se dispone de 10 sensores táctiles capacitivos en 10 GPIOs.

II. Formulación del Problema

El problema consiste en diseñar e implementar con el ESP32 un seguidor solar, para paneles solares, de un eje horizontal conocido técnicamente como 1xh.

El sistema diseñado estará conformado por los siguientes elementos y características:

- Una superficie plana de tamaño mínimo 20cmx30cm (en un material que soporte el trabajo a realizar), que simulará el panel solar a controlar.
- Un servomotor ajustado a una estructura mecánica para poder realizar el movimiento del seguidor solar, controlado por el PWM interno del ESP32.
- Un RTC (Real Time Clock) DS1307 o similar que opere con protocolo I2C.
- Una LCD TFT ILI9163 o similar.
- Al encender es sistema, se iniciará la hora del RTC a las 5:59am, junto con los datos de la fecha correspondiente al día de la entrega del proyecto, valores que serán visualizados constantemente en la LCD TFT.
- El sistema realizará el control del desplazamiento del panel durante 12 horas al día (se supone constante la cantidad de sol por día).

III. Diseño y Modelo de solución

Lo primero que se contemplo para la elaboración del proyecto son los elementos requeridos, en primer lugar se empleo 1 pantalla TFT de 128x128, en dicha pantalla se mostrara la información pertinente para el desarrollo del proyecto, como lo es la hora en tiempo real empleando el RTC, la la fecha actual, la velocidad con la que esta funcionando el sistema y la posición actual del panel.

El control del tiempo real es de vital importancia para el correcto funcionamiento del sistema ya que al considerarse que el sol cambia de posición y que el sistema debe ser capaz de seguirlo según la hora del día , se requiere de un elemento externo capaz de brindarle esta información al ESP32 con la cual se pueda progrmar las instrucciones requeridas.

La operación del seguidor solar a implementar será la siguiente:

- Cuando sean las 6:00am el sistema iniciará el control del movimiento de manera automática, esta tarea se

realizará durante 12 horas, al llegar a las 6:00pm el sistema detendrá el control y retornara el panel a la posición inicial a una velocidad de 3°/m, de acuerdo a la última velocidad seleccionada, permitiendo repetir el proceso diariamente.

- Se tendrán cuatro velocidades de desplazamiento del panel solar: 15°/h (1° cada 4 min.), 12°/h (1° cada 5 min.), 10°/h (1° cada 6 min.) y 6°/h (1° cada 10 min.). El valor por defecto para la velocidad será 12°/h, debiendo ubicar el panel en la posición inicial 18°.

Para el control de las velocidades, se implementaron los sensores capacitivos, cada uno de estos esta asociado a una velocidad diferente, por lo que al pulsarlos se tendrá una posición inicial distinta, así como el desplazamiento del panel se realizara de una forma mas rápida o mas lenta, para implementar estos sensores se propone tomar muestras y hacer un promedio , con lo que se determina un umbral se activación para cada sensor, esto permite que los sensores no se activen de forma errónea.

Para el control de desplazamiento cada determinado tiempo se utilizo un DS1307 RTC bajo el protocolo I2C. Usando la libreria ”RTClib.h” de arduino fue posible establecer tanto el intervalo de tiempo como la fecha deseada. A partir del objeto ”now()” de la librería mecionada anteriormente el cual cuenta internamente con la función ”timestamp” que nos permite obtener tanto la fecha como la hora actual del reloj DS1307 RTC. En base a esto se pudo realizar la visualización en la TFT y la comparación correspondiente al intervalo de tiempo final del panel solar.

Posterior a esto se emplea la librería ”ESP32Servo.h”, con la cual por medio de la función **servoMotor.write()** podemos implmentar un PWN y desplazar el servomotor a la posición deseada.

Para el control de la pantalla se empleo la comunicación SPI, en la cual se emplearon las conexiones se aprecia en la figura 1.

Figure 1: Conexión SPI

Para esta conexión es importante considerar, que por defecto en la comunicación SPI se tiene una velocidad de 8 MHz.

A. *Pines Empleados*

Para el control de los sensores capacitivos, se requieren un total de 4 pines para esta tarea, para el control del servomotor se requiere de 1 pin, para el RTC son requeridos 2 pines mas, finalmente para la pantalla TFT empleando comunicación SPI se requieren 4 pines, por lo que en total se requerirán de 8 pines para el desarrollo de la practica, estos pines se encuentran resumidos en la tabla I.

Tarea	#Pin	Funcionalidad
Control RTC	GPIO22	SCL
	GPIO21	SDA
Servo Motor	GPIO 26	Control Desplazamiento
Control de Pantalla TFT	GPIO5	CS
	GPIO3	DC
	GPIO23	MOSI
	GPIO18	SCK
Control de Sensores Capacitivos	GPIO32	Control de velocidad en 15°/h (Sensor T9)
	GPIO12	Control de velocidad en 12°/h (Sensor T5)
	GPIO14	Control de velocidad en 10°/h (Sensor T6)
	GPIO27	Control de velocidad en 6°/h (Sensor T7)

Table I: Distribucón Pines Empleados

Posteriormente se procedió a desarrollar e implementar con la ayuda del entorno de desarrollo de arduino IDE el código adecuado para el funcionamiento del seguidor solar y el cual podemos apreciar en el anexo 1.

IV. Resultados

Se desarrollo el siguiente montaje :

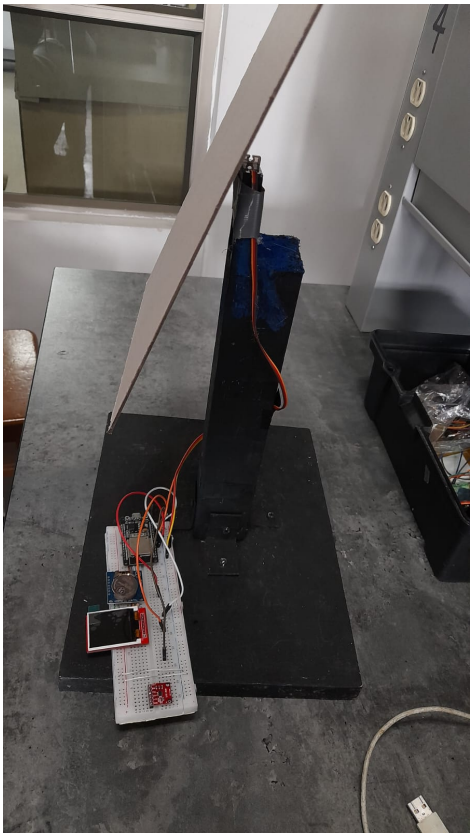

Figure 2: Montaje Final

En la figura 2 se aprecia el montaje final realizado.

V. Conclusiones

- Es necesario establecer un rango de operación para los sensores capacitivos ya que estos tienden a ser muy sensibles al ruido y en consecuencia afectar el comportamiento del sistema.
- Gracias a la implementación de los timers como interrupciones podemos generar un conjunto de diferentes acciones periódicas. Así mismo, podemos observar que gracias a las librerías de arduino es posible modificar dichas interrupciones de una forma mas rápida y eficiente.
- Gracias a que esta tarjeta tiene la posibilidad de ser programada en el entorno de desarrollo correspondiente a Arduino IDE el cual utiliza C como lenguaje de programación y nos facilita diversas librerías que son de gran utilidad al momento de usar diferentes características o componentes externos como por ejemplo la pantalla TFT y su implementación de la comunicación SPI, etc.
- La implementación del protocolo I2C para la transmisión de datos, en este caso para el control del RTC, facilita la tarea, ya que solo se requieren dos líneas de bus; una línea de datos en serie (SDA) y una línea de reloj en serie SCL, esto resulta muy útil a la hora de incorporar nuevos módulos en un montaje físico.
- Comprender el manejo de los sensores capacitivos en el ESP32 es de vital importancia para el desarrollo del proyecto, puesto que es necesario considerar los umbrales requeridos para su activación y que no se activen en vació, ya que en caso de tener un umbral muy bajo, se activaran los sensores en momentos erróneos, así como en caso de presentarse umbrales muy altos, el sensor no sera capaz de registrar las pulsaciones y no se verán reflejadas las interacciones el funcionamiento del programa, adicional a esto es necesario que los cables implementados para estos sensores no queden al aire y no se muevan en lo posible, para evitar falsas activación.

VI. Anexos

```
#include <ESP32Servo.h>
#include "RTClib.h"

#include <SPI.h>
#include <TFT_ILI9163C.h>
#include <Adafruit_GFX.h>



#define SERVO_PIN 26  // ESP32 pin GIOP26 connected to servo motor
#define LED 2
#define CS 5 //pin CS0
#define DC 3 //pin A0
#define MOSI 23 //pin SDA (predefinido en la libreria SPI)
#define SCK 18 //pin CLK (predefinido en la libreria SPI)


TFT_ILI9163C tft=TFT_ILI9163C(CS,DC); //Se crea el objeto de nombre "tft"

//Definici n de colores en formato RGB565
#define BLACK   0x0000
#define WHITE   0xFFFF
#define RED     0xF800 //0b1111100000000000
#define GREEN   0x07E0 //0b0000011111100000
#define BLUE    0x001F //0b0000000000011111
#define YELLOW  0xFFE0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define ORANGE  0xFD20


uint16_t umbralSensorT9 = 0;
uint16_t umbralSensorT5 = 0;
uint16_t umbralSensorT6 = 0;
uint16_t umbralSensorT7 = 0;
uint8_t SensorT9, SensorT5, SensorT6, SensorT7;
int segundo, minuto, hora, dia, mes, anio;
int velocidad;
int velAct;
uint8_t pos = 80;
char date[10] = "06:00:10";
DateTime now;

RTC_DS1307 rtc;
Servo servoMotor;
hw_timer_t *My_timer = NULL;

void IRAM_ATTR onTimer() {
  // digitalWrite(LED, !digitalRead(LED));

//   if(digitalRead(LED)==HIGH){
//     if(pos>18){
//       if(pos<=21){
//         pos = 18;
//       }else{
//         pos=pos-3;
//       }
//     }else{
//       if(pos>=15){
//         pos = 18;
//       }else{
//       pos = pos+3;
//       }
//     }
//   }else{
//     if (pos < 180) {
//       pos++;
//     }
//   }
  if(digitalRead(LED)==HIGH){
```

```
    if (pos > 18) {
       pos-=3;
    }else{
      pos = 18;
      servoMotor.write(pos);
    }
  }else{
    if (pos < 180) {
       pos++;
    }
  }
  servoMotor.write(pos);
}

void setUmbral() {
  for (uint8_t i = 0; i < 200; i++) {
    umbralSensorT9 += touchRead(T9);
    umbralSensorT5 += touchRead(T5);
    umbralSensorT6 += touchRead(T6);
    umbralSensorT7 += touchRead(T7);
    delay(1);
  }
  umbralSensorT9 /= 200;
  umbralSensorT5 /= 200;
  umbralSensorT6 /= 200;
  umbralSensorT7 /= 200;

  umbralSensorT9 *= 0.5;
  umbralSensorT5 *= 0.5;
  umbralSensorT6 *= 0.5;
  umbralSensorT7 *= 0.5;
}

void updateTime(uint32_t seg) {
  timerAlarmWrite(My_timer, seg * 10000, true);
  // print in display
}

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  tft.begin(); //Inicializar la LCD
  tft.setTextColor(BLACK,CYAN);
  tft.fillScreen(CYAN);

  if (!rtc.begin()) {
   // Serial.println("Couldn't find RTC");
    Serial.flush();
    while (1) delay(10);
  }
  DateTime now = rtc.now();
  if (!rtc.isrunning()) {
    //Serial.println("RTC is NOT running, let's set the time!");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  } else {

    rtc.adjust(DateTime(now.year(), now.month(), now.day(), 5, 59, 0));
  }
  tft.setCursor(10,10);
  tft.print(now.year());



  My_timer = timerBegin(0, 8000, true);
  timerAttachInterrupt(My_timer, &onTimer, true);
  updateTime(5);
  timerAlarmEnable(My_timer);  //Just Enable
  velocidad=12;

  setUmbral();
  tft.setCursor(15,40);
```

```
  tft.print("velocidad: ");
  tft.setCursor(10,50);
  tft.print("Posicion");
  Serial.println("Umbrales");
  Serial.println(umbralSensorT9);
  Serial.println(umbralSensorT5);
  Serial.println(umbralSensorT6);
  Serial.println(umbralSensorT7);

  servoMotor.attach(SERVO_PIN);  // attaches the servo on ESP32 pin
  servoMotor.write(pos);
}

void loop() {
  DateTime time = rtc.now();
  tft.setCursor(10,20);
  tft.print(time.timestamp(DateTime::TIMESTAMP_TIME));
  tft.setCursor(10,30);
  tft.print(time.timestamp(DateTime::TIMESTAMP_DATE));

  tft.setCursor(70,50);
  if (pos<10){
    tft.print("  ");
    tft.print(pos);
  }else if(pos<100){
    tft.print(" ");
    tft.print(pos);
  }else{
    tft.print(pos);
  }
  if (velocidad!=velAct){
    tft.setCursor(75,40);
    if((velocidad/10)>=1){
      tft.print(velocidad);
    }else{
      tft.print(" ");
      tft.print(velocidad);
    }
    tft.print((char)248);
    tft.print("/h");
     velAct=velocidad;
  }
    if(time.timestamp(DateTime::TIMESTAMP_TIME)<date){
      digitalWrite(LED, LOW);
      SensorT9 = touchRead(T9);
      SensorT5 = touchRead(T5);
      SensorT6 = touchRead(T6);
      SensorT7 = touchRead(T7);
      Serial.print("SensorT9 ");
      Serial.println(SensorT9);

    if (SensorT9 < umbralSensorT9 && SensorT9 > 5) {
      SensorT9 = touchRead(T9);
      if (SensorT9 < umbralSensorT9 && SensorT9 > 5){
        updateTime(4);
        //pos = 0;
        //servoMotor.write(pos);
        velocidad=15;
      }
      while (SensorT9 < umbralSensorT9) { SensorT9 = touchRead(T9); }
      delay(200);
    }
    if (SensorT5 < umbralSensorT5 && SensorT5 > 5 ) {
      updateTime(5);
      //pos = 18;
      //servoMotor.write(pos);
      while (SensorT5 < umbralSensorT5) { SensorT5 = touchRead(T5); }
      velocidad=12;
      delay(200);
    }
    if (SensorT6 < umbralSensorT6 && SensorT6 > 5) {
```

```
213         updateTime(6);
214         //pos = 30;
215         //servoMotor.write(pos);
216         while (SensorT6 < umbralSensorT6) { SensorT6 = touchRead(T6); }
217         velocidad=10;
218         delay(200);
219       }
220       if (SensorT7 < umbralSensorT7 && SensorT7 > 5) {
221         updateTime(10);
222         //pos = 54;
223         //servoMotor.write(pos);
224         while (SensorT7 < umbralSensorT7) { SensorT7 = touchRead(T7); }
225         velocidad=6;
226         delay(200);
227       }
228     }else{
229       updateTime(1);
230       velocidad = 3;
231       digitalWrite(LED,HIGH);
232     }
233 
234 }
```

Listing 1: Código Proyecto 3