"MANEJO DE PERIFÉRICOS DE E/S (Teclado Matricial y barrido de displays)"

Trabajo Preparatorio N°6
Laboratorio de Sistemas Embebidos

Melanny Cecibel Dávila Pazmiño
Ingeniería en Telecomunicaciones
Facultad de Eléctrica y Electrónica
Quito, Ecuador
melanny.davila@epn.edu.ec

***Abstract*—En el siguiente preparatorio se va presentar la forma en la que se pueden usar los puertos analógicos como entradas o salidas digitales. Además, se van a establecer algunos parámetros para el uso de un barrido de displays de 7 segmentos y el manejo de periféricos de E/S.**

***Index Terms*—Arduino, periféricos, teclado matricial, display.**

I. Introducción

Un periférico de entrada/salida o E/S es aquel tipo de dispositivo periférico de un dispositivo capaz de interactuar con los elementos externos a ese sistema de forma bidireccional, es decir, que permite tanto que sea ingresada información desde un sistema externo, como emitir información a partir de ese sistema [1].

El teclado matricial es un dispositivo bastante común en proyectos con sistemas microcontrolados como el Arduino y está conformado básicamente por un arreglo de pulsadores distribuidos en forma de matriz, que puede ser leído utilizando pocos pines del Arduino y utilizando el concepto de la multiplexación. En el mercado podemos encontrar Teclados matriciales de diferentes tamaños, como el teclado matricial 4×4, 3×3, 4×3, entre otros. Estos teclados pueden tener una estructura de membrana o una estructura rígida [1].

II. Objetivos

- Relacionar al estudiante con el uso y manejo de periféricos de entrada y salida en Arduino
- Conocer acerca de las diferentes librerías para el control de periféricos de entrada o salida en Arduino [2].

III. Cuestionario

A. Consultar sobre el funcionamiento de las entradas analógicas de Arduino Uno como entradas y salidas digitales. Describir las configuraciones necesarias.

Los pines analógicos se pueden usar de manera idéntica a los pines digitales, utilizando los alias A0 (para la entrada analógica 0) hasta A5 en el caso de Arduino UNO. Se caracterizan por leer valores de tensión de 0 a 5 Voltios con una resolución de 1024 (10 bits). Si se divide 5 entre 1024 se tendrá que es capaz de detectar variaciones en el nivel de la señal de entrada de casi 5 mV [1].

Para hacer la lectura de uno de estos pines se escribe:

$Pin1 = analogRead(Pin);$

Donde: "Pin1" es el nombre de la variable donde se quiera almacenar el valor leído y en "Pin" se tendrá que para poner el número del pin analógico que se ha elegido (0-5) o el nombre de la variable que almacena dicho número. Esta función devolverá un valor que va de 0 a 1023 en proporción al nivel de la señal de entrada. Para una entrada nula obtendremos el valor 0, para una entrada de 2.5 Voltios 511 (la mitad de 1023) y para 5 Voltios 1023. [1]

Los pines analógicos también tienen resistencias pullup, que funcionan de manera idéntica a las resistencias pullup en los pines digitales. Se habilitan con el siguiente "comandopinMode(A0, INPUT-PULLUP)" .

El comando analogRead() no funcionará correctamente si un pin se ha configurado previamente en una salida, por lo que si este es el caso, vuelva a configurarlo en una entrada antes de usar analogRead. Del mismo modo, si el pin se ha configurado en ALTO como salida, la resistencia pull-up se configurará cuando se vuelva a conectar a una entrada. Además, se pueden usar los pines analógicos como entradas digitales con el comando digitalRead(), este comando redondeará los valores recibidos para traducirlos a 1L o 0L.

Las entradas digitales son las mismas que las salidas digitales, es decir, los pines que van del 1 al 13. Se diferencian de las analógicas porque éstas son capaces de "entender" sólo dos niveles de señal, LOW o valores cercanos a 0 V y HIGH o valores cercanos a 5 V [1].Aunque los pines digitales por defecto vienen configurados como entradas, si se quiere hacerlo manualmente se escribe:

pinMode(Pin,INPUT);

Para almacenar los dos valores posibles LOW o HIGH en una variable llamada "lectura" se escribe:

lectura = digitalRead(Pin);

Todos los pines digitales de Arduino pueden actuar como salidas digitales (por ello se denominan I/O, input y output). Pero conviene destacar que los pines analógicos también pueden usarse como entradas y salidas digitales.

B. Determinar el rango de frecuencias para el barrido de un arreglo de 4 displays de cátodo/ánodo común para que se visualice de manera adecuada.

El rango de frecuencias apropiado para que no se note el parpadeo de los LEDs mientras se hace la multiplexación se encuentra entre los 60 [Hz] hasta 1 [kHz]. Un dispositivo que se puede tomar en cuenta como ejemplo de este rango de frecuencias es un foco que se prende y apaga a la frecuencia de la energía eléctrica que es enviada a los hogares (60[Hz]).

Se recomienda que se tome como base esta frecuencia y que se multiplique 60 por el número de displays de 7 segmentos que se tiene en el circuito para obtener la frecuencia óptima para realizar un barrido de displays apropiado [2].

C. Consultar el funcionamiento y distribución de pines PWM en Arduino Uno y Arduino Mega. Describir las librerías necesarias y parámetros de configuración para la generación de señales PWM.

La modulación de ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales. El control digital se utiliza para crear una onda cuadrada, una señal conmutada entre encendido y apagado. Este patrón de encendido y apagado puede simular voltajes entre encendido total (5 voltios) y apagado (0 voltios) al cambiar la parte del tiempo que la señal pasa en encedido en comparación con el tiempo que la señal pasa en apagado.

La duración del tiempo encendido se llama ancho de pulso. Para obtener valores analógicos variables, puede cambiar o modular ese ancho de pulso. Si repite este patrón de encendido-apagado lo suficientemente rápido con un LED, por ejemplo, el resultado es como si la señal es un voltaje constante entre 0 y 5v que controla el brillo del LED [3].

En la figura 1 a continuación, las líneas verdes representan un período de tiempo regular. Esta duración o período es el inverso de la frecuencia PWM. En otras palabras, con la frecuencia PWM de Arduino a aproximadamente 500Hz, las líneas verdes medirían 2 milisegundos cada una.

Una llamada a analogWrite() est´a en una escala de 0 a 255, de modo que analogWrite(255) solicita un ciclo de trabajo del 100% (siempre activado) y analogWrite(127) es un ciclo de trabajo del 50% (la mitad del tiempo) para ejemplo.[3]

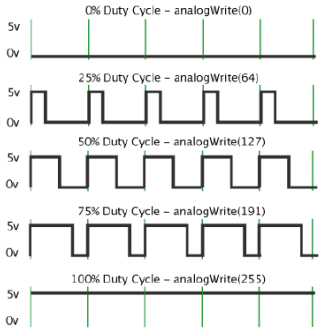

Fig. 1. PWM

Las librerías usadas para esta modulaci´on son: $\#include < PWM.h >$ y la distribuci´on de los pines es la siguiente:

Fig. 2. Distribución de pines

D. Se tiene una caja fuerte la cual funciona con un código de acceso de 4 dígitos, para el ingreso de los dígitos se utiliza un teclado matricial, la primera vez que el usuario utilice la caja fuerte deberá establecer la clave de funcionamiento. Posteriormente si el usuario ingresa la clave correcta, un servomotor se moverá de 0 a 90°simulando la apertura de la caja fuerte y en el monitor serial se desplegará el mensaje "CLAVE CORRECTA". Por el contrario, si el usuario ingresa la clave incorrecta el servomotor no se moverá y en el monitor serial se mostrará el número de intentos que quedan antes que la caja fuerte se bloquee. El número de intentos máximo es 3. Cuando el usuario supere el número de intentos máximos la caja fuerte se bloqueará por 30 segundos, posterior a lo cual el usuario tendrá otros 3 intentos para ingresar la clave. Se recomienda emplear las librerías "Keypad.h" para el barrido de teclado y "Servo.h" para el uso del servomotor.

A continuación, se presenta el esquemático realizado en el software de simulación Tinkercad junto con el código fuente que permite registrar una clave de ingreso a una caja fuerte.

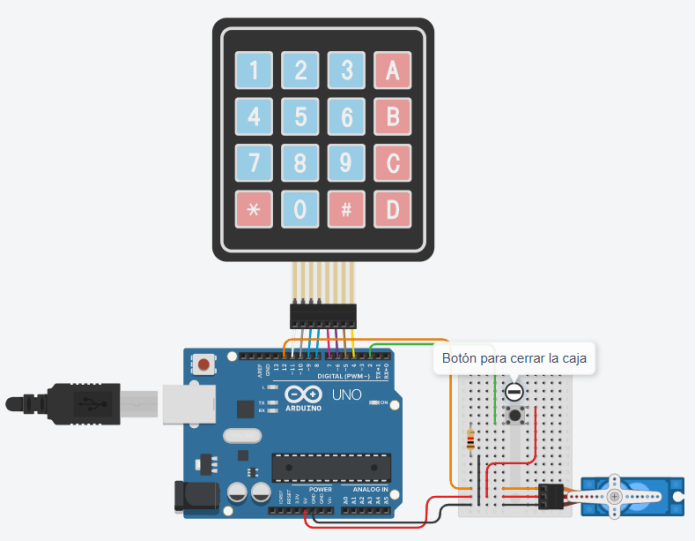

Fig. 3. Esquemático implementado

```
#include <Keypad.h>
//Servomotor
#include <Servo.h>
//Variables para el programa
//Variable del servo
Servo Motor;
//Varibles para el teclado
//Numero de filas y columnas
const byte Filas = 4;
const byte Colm = 4;
//Pines para filas y columnas
byte Filas_P[] = {11,10,9,8};
byte Colm_P[] = {7,6,5,4};
//Se define la estructura de nuestro teclado.
char Teclas [ Filas ][ Colm ] =
{
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
//Se crea el teclado
Keypad Teclado = Keypad(makeKeymap(Teclas),
    Filas_P, Colm_P, Filas, Colm);
//Variables para comprobar la clave
int existe = 0;
int contador = 0;
int intentos = 0;
char clave[4];
char claveingresada[4];
//Variable para saber si la caja esta abierta o
    no
int abierto = 0;
void setup()
{
//Se inicia el Serial
Serial.begin(9600);
//Declaramos el pin para el Servomotor
Motor.attach(12);
//Se lo pone en 0
Motor.write(0);
//Interupcion para cerrar la caja
attachInterrupt(digitalPinToInterrupt(2),Cerrar,
    RISING);
}
//Funcion Crearclave
//Crea la clave si se inicio el programa
void crearClave()
{
Motor.write(0);
//Indico que se va a crear una clave
Serial.println("Sistema de Caja Fuerte");
Serial.println("Primer Ingreso");
Serial.println("Ingrese 4 digitos para crear su
    clave");
while(contador < 4)
{
//variable para guardar la tecla presionada
char tecla = Teclado.getKey();
//Si se presiona la tecla la guardo como parte
    de la clave
if(tecla != 0)
{
clave[contador] = tecla;
contador++;
}
}
//Indico que se creo la clave
Serial.println("Clave creada");
Serial.println("La caja permanecera bloqueada
    hasta que ingrese la clave nuevamente");
Serial.println("");
Motor.write(0);
Serial.println("Caja cerrada");
Serial.println("");
contador = 0;
}
//Funcion Comprobar
//Comprueba si la clave ingresada es la correcta
int Comprobar()
{
//control indica si la clave se ingreso
    correctamente
//0 -> Correcto
//1 -> Incorrecto
int control = 0;
for(int i=0;i<4;i++)
{
if(clave[i] != claveingresada[i])
{
control = 1;
break;
}
}
return control;
}
//Funcion Clave
//Arma la clave con las pulsaciones del teclado
void Clave()
{
while(intentos < 3)
{
Serial.println("BIENVENIDO");
Serial.println("Ingrese su clave: ");
  while(contador < 4)
{
//variable para guardar la tecla presionada
char tecla = Teclado.getKey() ;
//Si se presiona la tecla la guardo como parte
    de la clave
if(tecla != 0)
{
claveingresada[contador] = tecla;
contador++;
}
}
//Compruebo si la clave es correcta
if(Comprobar() == 0)
{
//De ser correcta abro la caja
abierto = 1;
//Indico que se ingreso con exito
```

```
Serial.println("CLAVE CORRECTA");
Serial.println("Caja abierta");
Serial.println("");
//Muevo el Servomotor
Motor.write(90);
//Desactivo alerta de movimiento
detachInterrupt(digitalPinToInterrupt(3));
//Activo boton para cerra la caja
attachInterrupt(digitalPinToInterrupt(2),Cerrar,
    RISING);
contador = 0;
break;
}
else
{
//Si falla aumento la cuenta de intentos
intentos++;
//Indico que fallo
Serial.println("CLAVE INCORRECTA");
//Indico los intentos restantes
Serial.print("Intentos sobrantes: ");
Serial.print(3-intentos);
Serial.println("");
//Reinicio el contador para el ingreso
contador = 0;
}
}
}
//Funcion Loop principal
void loop()
{
while (abierto == 0)
{
//Desactivo la interrupcion que cierra la caja
detachInterrupt(digitalPinToInterrupt(2));
  //Comprobamos si la clave existe
if (existe == 0)
{
crearClave();
existe = 1;
}
//Recogo la clave de ingresada para comprobarla
Clave();
//Compruebo el numero de intentos para ingresar
if(intentos == 3)
{
Serial.println("Intentos agotados, debe esperar
    30 segundos");
Serial.println("");
//Espera de 30 segundos si se ingresa mal 3
    veces
delay(5000);
//Reinicio lo intentos
intentos = 0;
Serial.println("Puede volver a ingresar la clave
    ");
Serial.println("");
}
}
}
//ISR cerrar
//Simulo que se cierra la caja
void Cerrar()
{
abierto = 0;
intentos = 0;
//Cierro la caja fuerte
Serial.println("Caja cerrada");
Serial.println("");
Motor.write(0);
}
```

Código 1: Implementación clave de una caja fuerte

REFERENCES

[1] "Teclado Matricial con ARDUINO- [4x4, 4x3 y Cualquier TECLADO]". https://controlautomaticoeducacion.com/arduino/teclado-matricial-keypad/ (accedido ene. 13, 2021).

[2] E. Espinosa, E. Tatayo, "MANEJO DE PERIFÉRICOS DE E/S (Teclado Matricial y barrido de displays)". C.P. SISTEMAS EMBEBIDOS, Accedido: ene. 12, 2020. [En línea].

[3] G. Pérez, "Tutorial Arduino: Entradas Analógicas y Digitales," Open Webinars , 22 Abril 2015. [En línea]. Available: https://openwebinars.net/blog/tutorial-arduino-entradas-analogicas-y-digitales/. (accedido ene. 11, 2021).

[4] "S.O. - Entrada / Salida". http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/SO5.htm (accedido ene. 10, 2021).

[5] Avinash, "Multiplexed Seven Segment Displays," 04 Octubre 2008. [En línea]. Available: https://extremeelectronics.co.in/avr-tutorials/multiplexed-seven-segment-displays/. (accedido ene. 11, 2021)..

[6] "Entrada/salida". http://www.sc.ehu.es/sbweb/fisica/cursoJava/ fundamentos/archivos/intro.htm (accedido ene. 13, 2021).