“MANEJO DE PÓRTICOS DE ENTRADA Y SALIDA EN ARDUINO”

Informe N°3

Laboratorio de Sistemas Embebidos

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

2nd Jonathan Álvarez
Ingeniería en Telecomunicaciones
Facultad de Elétrica y Eléctronica
Quito, Ecuador
jonathan.alvarez@epn.edu.ec

***Abstract*—En el siguiente documento se presentarán las ventajas y desventajas del uso de registros del microcontrolador en lo programación de sketches, con algunas implementaciones con el fin de familiarizar al estudiante con este entorno.**

***Index Terms*—Arduino, puertos, pins.**

I. Introducción

software libre, flexible y fácil de utilizar para creadores y desarrolladores de proyectos. Mediante el uso de IDE de Arduino es posible encontrar códigos que permiten declarar pines, como entrada o salida, así como la lectura de datos digitales o analógicos en la placa de Arduino ocupando los respectivos comandos.

II. Objetivos

- Relacionar al estudiante con el uso y manejo de los puertos de entrada y salida de la placa de desarrollo Arduino Uno.
- Establecer e identificar las características generales de la programación en Arduino.
- Diseñar e implementar códigos de programación que permitan al estudiante familiarizarse con esquemas de automatización [2].

III. Cuestionario

A. Realizar un cuadro comparativo de las ventajas y desventajas del uso de los registros del microcontrolador en la programación de sketches.

A continuación se presentan las principales ventajas y desventajas del uso de registros cuando se realiza programación de sketches.

TABLA I
VENTJAS Y DESVENTAJAS DEL USO DE REGISTROS

Ventajas	Desventajas
Al estar relacionados con lenguajes de niveles más bajo de programación permiten alterar cosas que no se podrían con el lenguaje de programación por defecto	Puede producir más errores de compilación
Pueden simplificar operaciones largas.	Requiere mayores conocimientos por parte del usuario
Su sintaxis no es muy complicada	Requiere que el usuario tenga un conocimiento general de lenguajes de menor nivel
Permiten un control más directo de los pines	Se pueden alterar funcionalidades importantes de la placa si no se usan adecuadamente

B. Programar en Tinkercad el cubo de leds 4x4x4 de la Figura 1 para que enciendan los leds, simulando los efectos especificados en la Tabla 2.

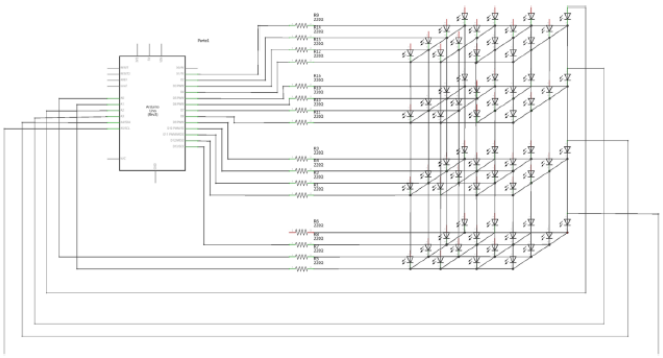

Fig. 1. Circuito esquemático matriz de LEDs 4x4

```
1  // 6.2 Informe
2 //El efecto especificado con el arreglo actual de
      los leds
3 // no es posible llevar a cabo el efecto deseado de
      manera exacta
4 //de todas maneras se ha recreado de mejor manera el
      efecto
```

TABLA II
ASIGNACIÓN DE EFECTO A REALIZAR.

GR1	Rain effect	https://www.youtube.com/watch?v=JT99QlEn3q8&ab_channel=chrmoe
GR2	Axis effect	https://www.youtube.com/watch?v=ujtFLRZjFww&ab_channel=chrmoe
GR3	Boing effect	https://www.youtube.com/watch?v=DT8YYW4Px1k&ab_channel=chrmoe
GR4	Sendvoxels rand z effect	https://www.youtube.com/watch?v=RUJN26cIGn4&ab_channel=chrmoe
GR5	Woopwoop effect	https://www.youtube.com/watch?v=yfP2Tx-Mzmo&ab_channel=chrmoe

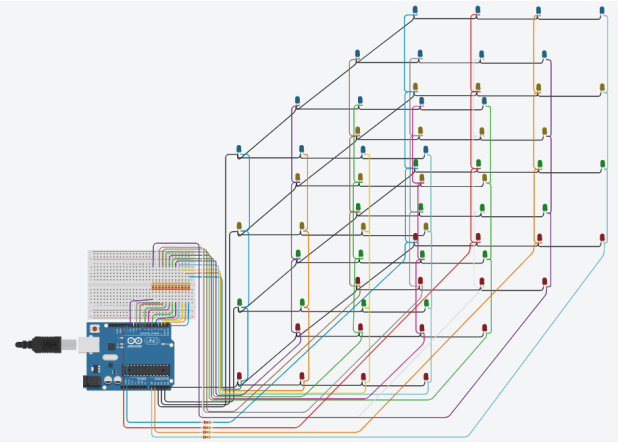

Fig. 2. Circuito controlador de un display de 7 segmentos mediante dos pulsadores

```

// Variable que controla las capas conectadas a tierra
int capa[4] = {A5, A4, A3, A2};
// Se controla una sola columna de leds y se incluye las variables
// de los pines analogos para que sea facil accesarlos
int col[16] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 , 11, 12, 13, A0, A1};

int i=0, rnd, var;//Variable aleatoria y acumulador
int t = 500;//Control del tiempo
void setup()
{
  //Serial.begin(9600);//Puerto serial para pruebas
  /*pinMode(A0,OUTPUT);
  pinMode(A1,OUTPUT);
  pinMode(A2,OUTPUT);
  pinMode(A3,OUTPUT);
  pinMode(A4,OUTPUT);
  pinMode(A5,OUTPUT);*/
  //Se definen todos los puertos como salidas
  DDRC = B11111111;
  DDRB = B11111111;
  DDRD = B11111111;
  //Se enceran todos los puertos para que los leds inicien apagados
  PORTC = B00111100;//Si estan en bajo permiten el paso de corriente
  PORTB = B00000000;
  PORTD = B00000000;

}
void apagar(){
  // Funcion para apagar todos los leds
  PORTC = B00111100;
  PORTB = B00000000;
  PORTD = B00000000;
}
void prender(){
  //Funcion para prender todos los leds
  PORTC = B00000011;
  PORTB = B11111111;
  PORTD = B11111111;

}
void sube(int r){
  // Ingresa un numero aleatorio correspondiente a la columna
  // Y se realiza efecto de luces hacia abajo
  digitalWrite(col[r],HIGH);
  digitalWrite(capa[0],LOW);
  delay(t);
  digitalWrite(capa[1],LOW);
  digitalWrite(capa[0],HIGH);
  delay(t);
  digitalWrite(capa[2],LOW);
  digitalWrite(capa[1],HIGH);
  delay(t);
  digitalWrite(capa[3],LOW);
  digitalWrite(capa[2],HIGH);
  delay(t);
  digitalWrite(capa[3],HIGH);
  digitalWrite(col[r],LOW);
}
void baja(int r){
  // Ingresa un numero aleatorio correspondiente a la columna
  // Se realiza el efecto de la luz moviendose hacia arriba
  digitalWrite(col[r],HIGH);
  digitalWrite(capa[3],LOW);
  delay(t);
  digitalWrite(capa[2],LOW);
  digitalWrite(capa[3],HIGH);
  delay(t);
  digitalWrite(capa[1],LOW);
  digitalWrite(capa[2],HIGH);
  delay(t);
  digitalWrite(capa[0],LOW);
  digitalWrite(capa[1],HIGH);
  delay(t);
  digitalWrite(capa[0],HIGH);
  digitalWrite(col[r],LOW);
}
void loop()
{
  //Realizamos un control para mostrar todos los leds funcionando
  prender();
  delay(t);
  apagar();
  delay(t);
  // Se eigio la condicion while par aaumentar facilmente las repeticiones
  //o para que se trate de un bucle infinito
  while(true){
    // Decide si la luz empezara arriba o abajo como se mostraba en el video
    rnd = random(2);//Decide si empieza arriba o abajo
    var = random(17);
    // A partir del primer numero aleatorio se decide si
    // el efecto empieza arriba o abajo
    switch(rnd){
      case 0:
```

```
        sube(var);//Escoge aleatoriamente el numero para subir
        break;
        case 1:
        baja(var);//Escoge aleatoriamente el numero para bajar
        break;
      }
      i++;
      if(i>50){
        break;
      }
    }
}
```

C. Simular un circuito que controle un display de 7 segmentos (0-F), utilizando dos pulsadores, uno para incremento y otro para decremento.

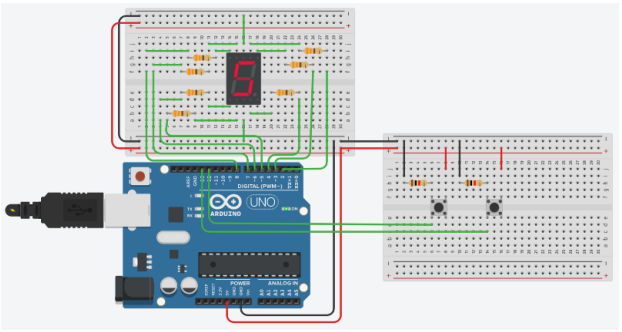

Fig. 3. Circuito controlador de un display de 7 segmentos mediante dos pulsadores

```
  //Encendido de display 7 Segmentos mediante ingreso de tecla del
//Usuario
int msj=0;
int i;
//Definimos los resultados
int seg[16] = {48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 97, 98, 99, 100, 101, 102};
int aum = 13;
int dis = 12;
int estaum, estdis;
void setup(); //Configuracion del monitor serial y puertos
int disp = 0;// Es el numero en el que empieza el display

void setup()
{
  Serial.begin(9600); // Se activa el puerto serial
  for(i=2; i<8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
  pinMode(aum, INPUT);
  pinMode(dis, INPUT);
}

void display(char a, int b, int c, int d, int e, int f, int g)
{
  digitalWrite(2,a);
  digitalWrite(3,b);
  digitalWrite(4,c);
  digitalWrite(5,d);
  digitalWrite(6,e);
  digitalWrite(7,f);
  digitalWrite(8,g);

}

void loop()
{
  //msj = Serial.read();
  estaum = digitalRead(aum);
  estdis = digitalRead(dis);

  if(estaum == HIGH){
    if(disp <15){
      disp++;
    }
  }
  if(estdis == HIGH){
    if(disp > 0){
      disp--;
    }
  }
  msj = seg[disp];
  switch (msj) {
  case 48:
    display(1,1,1,1,1,1,0);
    break;
   case 49:
    display(0,1,1,0,0,0,0);
    break;
   case 50:
    display(1,1,0,1,1,0,1);
    break;
   case 51:
    display(1,1,1,1,0,0,1);
    break;
   case 52:
    display(0,1,1,0,0,1,1);
    break;
   case 53:
    display(1,0,1,1,0,1,1);
    break;
   case 54:
    display(1,0,1,1,1,1,1);
    break;
   case 55:
    display(1,1,1,0,0,0,0);
    break;
   case 56:
    display(1,1,1,1,1,1,1);
    break;
   case 57:
    display(1,1,1,0,0,1,1);
    break;
   case 97:
    display(1,1,1,0,1,1,1);
    break;
   case 98:
    display(0,0,1,1,1,1,1);
    break;
   case 99:
    display(1,0,0,1,1,1,0);
    break;
   case 100:
    display(0,1,1,1,1,0,1);
    break;
   case 101:
    display(1,0,0,1,1,1,1);
    break;
   case 102:
    display(1,0,0,0,1,1,1);
    break;
  }
  delay(200);
```

```
}
```

D. Conclusiones:

Jonathan Álvarez

- Es posible ahorrar lineas de código mediante el uso de puertos para definir que pines serán de salida o entrada y sus estados iniciales.
- Es necesario optimizar el código debido a que existe la posibilidad de que para proyectos muy grandes la memoria del Arduino se termine por lo tanto es recomendable usar registros y puertos para asignacion rápida en pocas lineas de código.
- El bloque setup solo se ejecuta una vez a al iniciar el programa, por lo cual se debe en este bloque realizar la inicialización de las variables y definir que pines de entrada o sealida serán usados.

Melanny Dávila

- El uso de registros es una forma más simplificada de controlar los pines, sin embargo se debe tener un alto conocimiento a cerca de su conocimiento con el fin de utilizarlos de una manera adecuada.
- Los sistemas embebidos pueden llegar a ser una forma barata de implementación de sistemas de control de casi cualquier cosa que se desee automatizar.
- Se requiere un conocimiento básico de lenguajes de bajo nivel para poder usar de manera apropiada los registros y de esta manera se puede optimizar bastantes líneas de código.

E. Recomendaciones:

Jonathan Álvarez

- Usar nombres descriptivos de los pines en lugar de su numeración mediante la asignación en el principio del codigo a la variable descriptiva con el valor del pin, de esta manera el codigo será mas legible.
- Al tratar con asignaciones de pines digitales y analogicos al ser diferente su notacion se puede usar vectores que contengan las asignaciones en una variable de mismo tipo.

Melanny Dávila

- En el caso de no tener un conociento profundo sobre el uso de registros evitar su uso ya que se pueden cambiar configuraciones no deseadas de forma permanente.
- Colocar una salida en pull up o pull down con un capacitor en paralelo al pulsador para evitar evitar rebotes.

REFERENCES

[1] cdaviddav, Arduino Uno Tutorial [Pinout], DIYI0T. https://diyi0t.com/arduino-uno-tutorial/ (accedido dic. 10, 2020).

[2] E. Tatayo, "MANEJO DE PÓRTICOS DE ENTRADA Y SALIDA EN ARDUINO". C.P. SISTEMAS EMBEBIDOS, Accedido: dic. 11, 2020. [En línea].

[3] O. Lira, S. Hernández, y R. García, Manual de Programación Android-Arduino: Principios básicos de la programación móvil aplicados a entornos interactivos. Editorial Académica Española. OmniScriptum Publishing Group, 2017.

[4] O. Torrente, Arduino: Curso practico de formación, Alfaomega, 2013