“MANEJO DE PÓRTICOS DE ENTRADA Y SALIDA EN ARDUINO”

Trabajo Preparatorio N°3
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 trabajo preparatorio, se revisarán la distribución de puertos de la la placa Arduino Uno y las propiedades eléctricas y funcionales de los pines de entrada y salida. Además se revisará brevemente la sintaxis de las instrucciones necesarias para declarar un pin.**

***Index Terms*—Arduino, pin, analógico, digital.**

I. Introducción

Arduino es una plataforma de creación electrónica de código abierto, basada en hardware y 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..

III. Cuestionario

A. Consultar la distribución de los puertos de las placas Arduino Uno y Mega, propiedades eléctricas, características y funcionalidades.

Para la placa Arduino se tienen tres tipos de puertos:

- **Puertos digitales B (del 8 al 13):** se pueden configurar como entradas o salidas, por defecto vienen configurados como entradas de manera que no tienen que ser declarados como entradas con el comando pinMode(). Cuando están en esta configuración se colocan en modo de alta impedancia. Los pines de entrada no exigen mucho al circuito que están censando [3].
  En el caso de que se configure uno de estos pines como entrada y no se conecte nada a ellos se medirá a momentos ruido eléctrico del ambiente, o el acoplamiento capacitivo de estado del pin que está cerca. En el caso de estar configurados como salida con el comando pinMode() cambian su impedancia a un estado bajo. Esto significa que pueden proporcionar una cantidad sustancial de corriente a otros circuitos. Los pines Atmega pueden proveer corriente positiva o negativa hasta 40 [mA]. Se debe tener cuidado con los corto circuitos en los pines Arduino o al tratar corrientes altas entre sus pines ya que estos pueden destruir o dañar el transistor de salida en el pin o todo el chip Atmega [3].
- **Puertos analógicos de entrada:** pueden servir como convertidores A/D, el convertidor tiene una resolución de 10 bits retornando enteros entre 0 y 1023. Si bien la funci´on principal de los pines analógicos para la mayoría de los usuarios de Arduino es leer sensores analógicos, los pines analógicos también tienen toda la funcionalidad de uso general pines entrada/salida (GPIO) (los mismos que los pines digitales del 0 al 13). Los pines analógicos se pueden usar de manera idéntica a los pines digitales, utilizando los alias A0 (para la entrada analógica 0), A1, etc.
  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.
- **Puertos digitales D (del 0 al 7):** la modulación de ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales [3]. El control digital se utiliza para crear una onda cuadrada, una señal conmutada entre encendido y apagado. Este patrón de encendido-apagado puede simular voltajes entre encendido total (5 voltios) y apagado (0 voltios) cambiando la parte del tiempo que la señal pasa en comparación con el tiempo que la señal pasa. La duración de ”a tiempo”

se llama ancho de pulso.
Para obtener valores analógicos variables, cambia o modula 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. 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 [3].

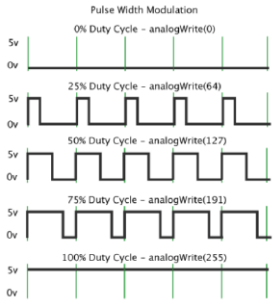

Fig. 1. Modulación por ancho de pulso

Arduino UNO

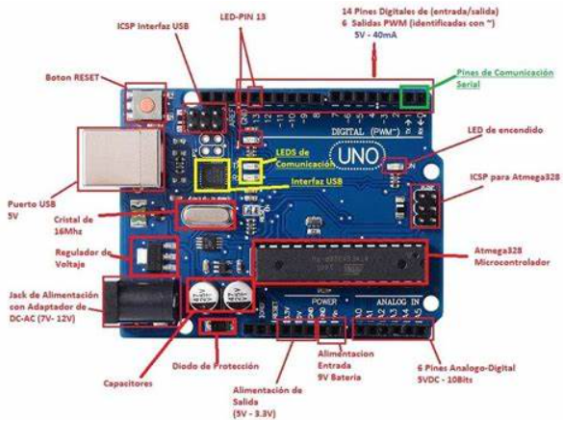

Fig. 2. Distribución de pines del Arduino UNO

Sus principales pines son:

- Pin de reset: tiene como función principal el de colocar en ceros el programa que se esté ejecutando en ese momento, para su funcionamiento se puede colocar un pulsador abierto con polaridad a tierra (GND), de manera que cuando el programa esté funcionando y se requiera volver a ceros, se pueda pulsar el botón y generar un cero a su entrada.
- Pin de referencia 3.3 voltios: es un voltaje generado por la placa.
- Pin de 5 voltios: se lo utiliza cuando la placa no puede ser alimentada externamente y se utilizaría una batería de 12 voltios para energizar los demás circuitos.
- Pin de tierra o GND: sirve para polarizar el circuito
- Entradas Analógicas: contiene a los pines A0, A1, A2, A3, A4 y A5 (analog in). Se los utiliza para que ingrese una señal de un sensor analógico, como un sensor de temperatura o un potenciómetro que, de un valor variable, también pueden ser utilizados como pines digitales.
- Pines Puerto Serial: son dos pines que son utilizados para entrada de señales tales como módulos de recepción de gps, bluetooth, video, etc.
- Pines Digitales de Entrada y Salida: se lo puede dividir a este bloque en varios segmentos como: Entradas y salidas digitales y Pines de salida analógicas.
- Puerto USB: este puerto tiene la función de cargar el programa en la memoria del Arduino, se encarga de la función de lectura y escritura en el IDE de Arduino, mas aun si se requieren que salgan datos que sean leídos desde un ordenador.
- Entrada de fuente regulada de 5 voltios: la tarjeta puede funcionar con un suministro externo de 6 a 20 voltios. Se puede suministrar menos de 7Vcc, sin embargo, si se alimenta con 5V, si la corriente es estable. Si se utiliza mas de 12V, el regulador se podría recalentar y la placa podría sufrir daños por los cual el rango recomendado es de 7 a 12V.

Arduino MEGA

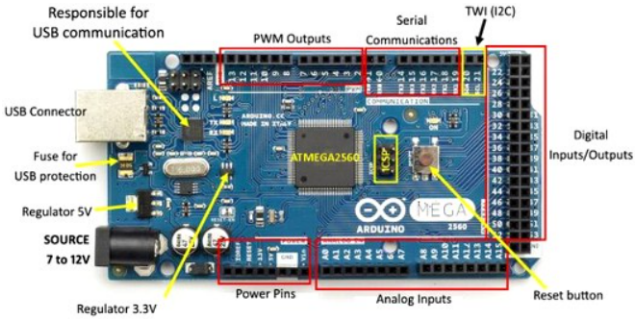

Fig. 3. Distribución de pines del Arduino MEGA

- Entrada de alimentación: La tensión de entrada a la placa cuando se utiliza una fuente de alimentación externa (en contraposición a 5 voltios de la conexión USB o de otra fuente de alimentación regulada). Se puede suministrar tensión a través de este pin, o, si el suministro de tensión es a través de la toma de alimentación, acceder a él a través de este pin.
- Conector de 5V: este pin es una salida de 5 V regulada del regulador de la placa. La placa puede ser alimentada ya sea desde el conector de alimentación de CC (7 - 12 V), por el conector USB (5 V), o por el pin Vin de la placa(7-12V).
- Pin de tierra o GND: sirve para polarizar el circuito

- IOREF: Este pin en la placa proporciona la referencia de tensión con la que opera el microcontrolador. Un escudo bien configurado puede leer la tensión del pin IOREF y seleccionar la fuente de alimentación adecuada o habilitar traductores de tensión en las salidas para trabajar con el 5 V o 3.3 V.
- Pines Digitales de Entrada y Salida: cada uno de los 54 pines digitales de la Mega se puede utilizar como una entrada o como una salida, utilizando las funciones pinMode(), digitalWrite() y digitalRead(). Operan a 5 voltios. Cada pin puede proporcionar o recibir 20 mA como condición de funcionamiento recomendada y tiene una resistencia de pull-up (desconectada por defecto) de 20-50 k ohmios.
- Entradas Analógicas: contiene 16 entradas analógicas, cada una de las cuales proporcionan 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto se miden de masa a 5 voltios, aunque es posible cambiar el extremo superior de su rango usando la función analogReference () y el pin AREF.

B. Describir los comandos y parámetros necesarios para la declaración de un pin y un puerto como entrada o salida tanto digital como analógico.

- **Entradas/Salidas Analógicas:**
  - analogReference() Configura la referencia de voltaje usada para la entrada analógica (es decir, el valor utilizado como la parte superior del rango de entrada) [4]. Las opciones son:
    - Predeterminado: la referencia analógica predeterminada de 5 voltios (en placas Arduino de 5 V) o 3,3 voltios (en placas Arduino de 3,3 V)
    - Interno: una referencia incorporada, igual a 1,1 voltios en el ATmega168 o ATmega328P y 2,56 voltios en ATmega32U4 y ATmega8 (no disponible en Arduino Mega)
    - Internal1V1: una referencia de 1.1V incorporada (solo Arduino Mega)
    - Internal2V56: una referencia incorporada de 2.56V (solo Arduino Mega)
    - Externo: el voltaje aplicado al pin AREF (0 a 5V solamente) se usa como referencia [4].
  - analogRead() Lee el valor del pin analógico especificado. Las placas Arduino contienen un convertidor de analógico a digital multicanal de 10 bits. Esto significa que asignará voltajes de entrada entre 0 y el voltaje de funcionamiento (5 V o 3,3 V) en valores enteros entre 0 y 1023. En un Arduino UNO, por ejemplo, esto produce una resolución entre lecturas de: 5 voltios / 1024 unidades o , 0,0049 voltios (4,9 mV) por unidad [5].
  - analogWrite() Escribe un valor analógico (onda PWM) a un pin especifico. No en todos los pines digitales se puede aplicar PWM.Se puede utilizar para encender un LED con distintos brillos o para impulsar un motor a distintas velocidades. Después de una llamada a analogWrite (), el pin generará una onda rectangular constante del ciclo de trabajo especificado hasta la próxima llamada a analogWrite () (o una llamada a digitalRead () o digitalWrite ()) en el mismo pin [6].
- **Entradas/Salidas Digitales:** Estos pines son los pines del 0 al 13 de Arduino y se llaman digitales porque sólo pueden manejar valores 0 o 1, para trabajar con ellos lo primero que se debe que hacer es configurar el modo de trabajo del pin. Ésto se hace siempre en la función setup() [7].

  Las instrucciones que se emplean para los pines digitales son:
  - pinMode(pin,[INPUT,OUTPUT]) Configura el modo de trabajo de pin digital, donde ”pin” es una variable con el valor correspondiente al número del pin a utilizar y se elige el modo de trabajo. Un pin digital tiene sólo dos modos, OUTPUT (salida) e INPUT (entrada). Si se declara un pin como OUTPUT, sólo podrá ser usado para activarlo, aplicando 5V en el pin, o para desactivarlo, aplicando 0V en el pin. Si se configura el pin como INPUT, sólo podrás usarlo para leer si hay 5V ó 0V en el pin [7].
  - digitalWrite(pin,valor) Se usa para activar o desactivar un pin digital. Entre paréntesis se debe indicar qué pin modificar, y qué valor darle. Ejemplo: digitalWrite(pin, HIGH);
    Ésto pondrá el pin en su estado HIGH, proporcionando 5V en él. Si se escribe LOW el pin será apagado, dejando el pin a 0V; hasta que se define el estado del pin como HIGH su valor por defecto será LOW [8].
  - digitalRead(pin); La instrucción digitalRead(pin) lee el estado de un pin y devuelve HIGH si está a 5V o LOW si hay 0V en él [9]. Para poder usar el valor del estado para algún fin debe ser guardado en una variable: Variable = digitalRead(pin);

C. Realizar ejemplos de lectura de un pin Digital mediante líneas de código en el aula de Tinkercad, utilizando:

A conitnuación se presentan los ejemplos realizados para la lectura de un pin digital con el uso de resistencias pull-up y pull-down.

- Resistencias de pull-up

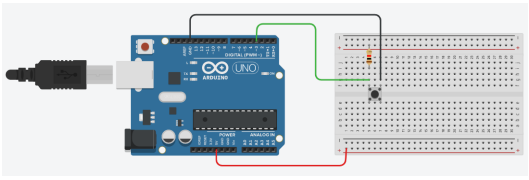

Fig. 4. Esquemático

- Resistencias de pull-down

```
/*Lectura de un Pin Digital/
/*EL siguiente código emite un mensaje de encendido en el caso
de detectar un 0L y en el caso de un 1L se emite un mensaje de
apagado*/

int value=0; //Creacion de una variable
int pin=3; //Variable que describe el numero de pin de entrada
int pin1=2; //Variable que describe el numero del pin de salida
int tiempo=1000; //Se crea la variable de tiempo (ms)
void setup(){
  pinMode(pin1, OUTPUT);//Se define al pin como salida
  pinMode(pin,INPUT_PULLUP);//Se define al pin como entrada
  Serial.begin(9600);//Se establece la velocidad de
  //recepción de datos
}
void loop(){
  value=digitalRead(pin);//lectura del pin de entrada

  if(value==HIGH){
    Serial.println("Encendido"); //impresion del mensaje
    digitalWrite(pin1,LOW);//se apaga el pin 2
  }else{
    Serial.println("Apagado");//impresion del mensaje
    digitalWrite(pin1,HIGH);//se enciende el pin 2
  }
  delay(tiempo);
}
```

Fig. 5. Código implementado

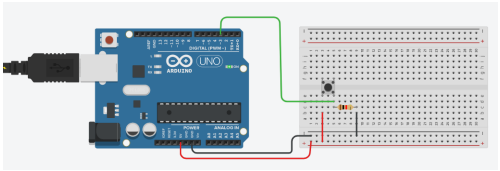

Fig. 6. Esquemático


```
/*Lectura de un Pin Digital/
/*EL siguiente código emite un mensaje de encendido en el caso
de detectar un 0L y en el caso de un 1L se emite un mensaje de
apagado*/

int value=0; //Creacion de una variable
int pin=3; //Variable que describe el numero de pin de entrada
int pin1=2; //Variable que describe el numero del pin de salida
int tiempo=1000; //Se crea la variable de tiempo (ms)
void setup(){
  pinMode(pin1, OUTPUT);//Se define al pin como salida
  pinMode(pin,INPUT_PULLUP);//Se define al pin como entrada
  Serial.begin(9600);//Se establece la velocidad de
  //recepción de datos
}
void loop(){
  value=digitalRead(pin);//lectura del pin de entrada

  if(value==HIGH){
    Serial.println("Encendido"); //impresion del mensaje
    digitalWrite(pin1,HIGH);//se apaga el pin 2
  }else{
    Serial.println("Apagado");//impresion del mensaje
    digitalWrite(pin1,LOW);//se enciende el pin 2
  }
  delay(tiempo);
}
```


Fig. 7. Código implementado

D. Realizar un ejemplo de escritura de un pin Digital mediante líneas de código en el aula de Tinkercad.

En las siguientes figuras se presenta un ejemplo que realiza la escritura de un pin digital

E. Realizar un ejemplo de lectura de un pin Analógico mediante líneas de código en el aula de Tinkercad.

Se presenta el ejemplo de lectura de un pin analógico a continuación.

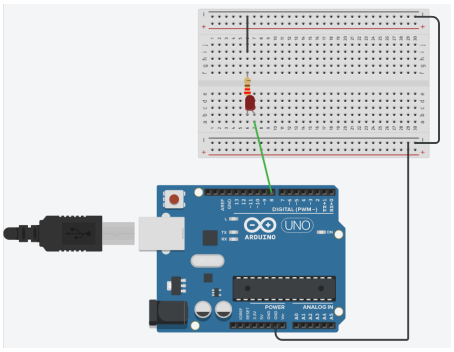

Fig. 8. Esquemático

```
/*Escritura de un Pin Digital*/
/*EL siguiente código permite encender y apagar un LED mediante
la escritura de un pin*/


int LED = 8; // Variable que describe el numero del pin
int tiempo=1000; //Se crea la variable de tiempo (ms)
void setup() {
  pinMode(LED, OUTPUT); //Se define al pin como salida
}

void loop() {
  digitalWrite(LED, HIGH);//Se enciende al LED
  delay(tiempo);//Tiempo de espera
  digitalWrite(LED, LOW);//Se apaga al LED
  delay(tiempo);//Tiempo de espera
}
```

Fig. 9. Código implementado

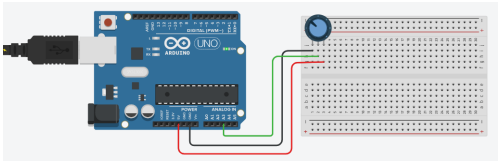

Fig. 10. Esquemático

```
/*Lectura de un Pin Analógico*/
/*EL siguiente código lee la salida de voltaje entre dos
terminales de un potenciometro*/

int entradaDeDatos; //Se crea una variable para leer los datos
float voltajeMedido;//Se crea una variable para
//usar el monitor serial
void setup()
{
  Serial.begin(9600);//Se establece la velocidad de
  //recepción de datos
}
void loop() {
  entradaDeDatos = analogRead(A3);//Lectura de los datos del
  //pin A3
  voltajeMedido = entradaDeDatos*(5.0 / 1023.0);//Mide el
  //voltaje en el potenciometro
  Serial.println(voltajeMedido);//Muestra el voltaje del
  //potenciometro
}
```

Fig. 11. Código implementado

Los archivos que se generarán responderán al siguiente nombre:" Apellido-Nombre-Prep3-Ejx-Gry", dónde "x" representa el número de ejercicios y "y" representa el grupo al que pertenecen.

REFERENCES

[1] Electrónica main “Arduino uno y la función de sus pines.”. Disponible en: https://www.electronicamain.com/arduino-uno-y-la-funcion-de-sus-pines/

[2] E. Tatayo, “MANEJO DE PÓRTICOS DE ENTRADA Y SALIDA EN ARDUINO”. C.P. SISTEMAS EMBEBIDOS, Accedido: dic. 09, 2020. [En línea].

[3] “Arduino - PortManipulation.” https ://www.arduino.cc/en/Reference/PortManipulation (accedido dic. 09, 2020).

[4] “analogReference() - Arduino Reference”. https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/ (accedido dic. 09, 2020).

[5] “analogRead() - Arduino Reference”. https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/ (accedido dic. 09, 2020).

[6] “analogWrite() - Arduino Reference”. https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/ (accedido dic. 09, 2020).

[7] “Entradas Digitales”, Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/tag/entradas-digitales/ (accedido dic. 09, 2020).

[8] “EntradaSalidaDigital.pdf”. [En línea]. Disponible en: http://platea.pntic.mec.es/ mhidalgo/documentos/07_EntradaSalidaDigital.pdf (accedido dic. 09, 2020).

[9] “digitalRead() - Arduino Reference”. https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/ (accedido dic. 09, 2020).