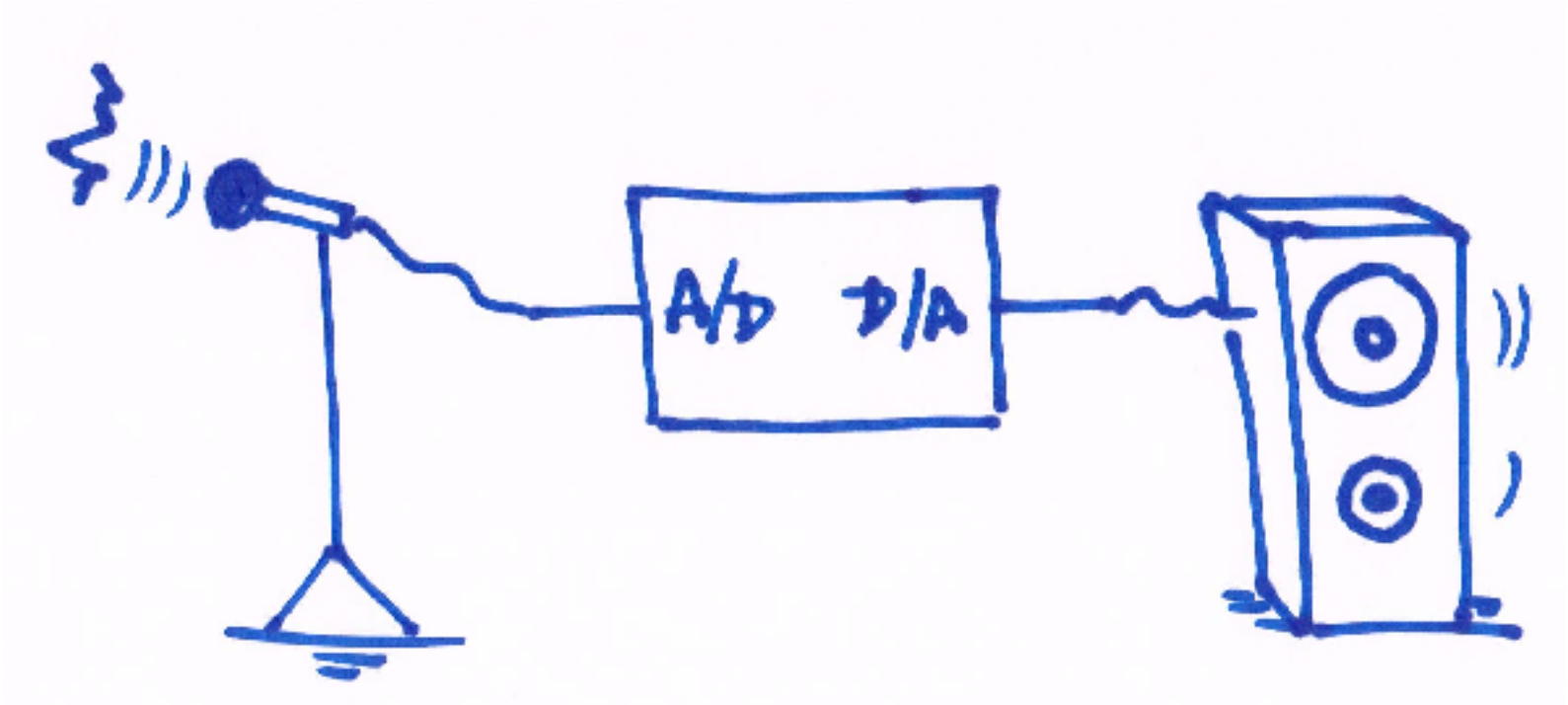
2. Programación en Arduino

Tallerine Arduino/LED

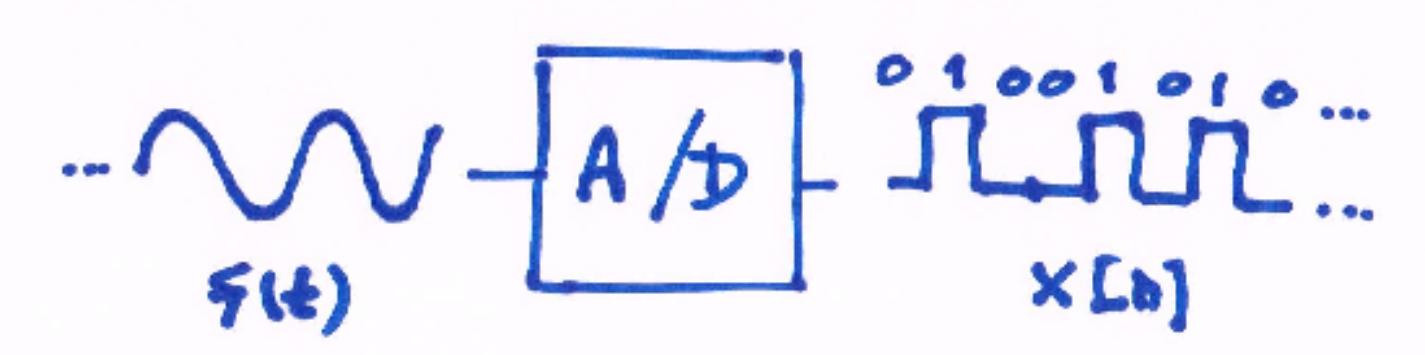
Instituto de Ingenería Eléctrica

12 de agosto de 2019

Entrada y salida **analógica**



Entradas analógicas: conversor A/D



- Las entradas analógicas se leen en los pines A0 a A5
- Sólo sirven para leer, no para escribir
- Los datos son leídos como números mediante un Conversor A/D:
  - Conversor Analógico/Digital (A/D) de 10 bits: recibe señales continuas entre 0 y 5V y retorna números enteros entre 0 y 1023
- la función `analogRead(int pin)` se usa para leer un dato analógico

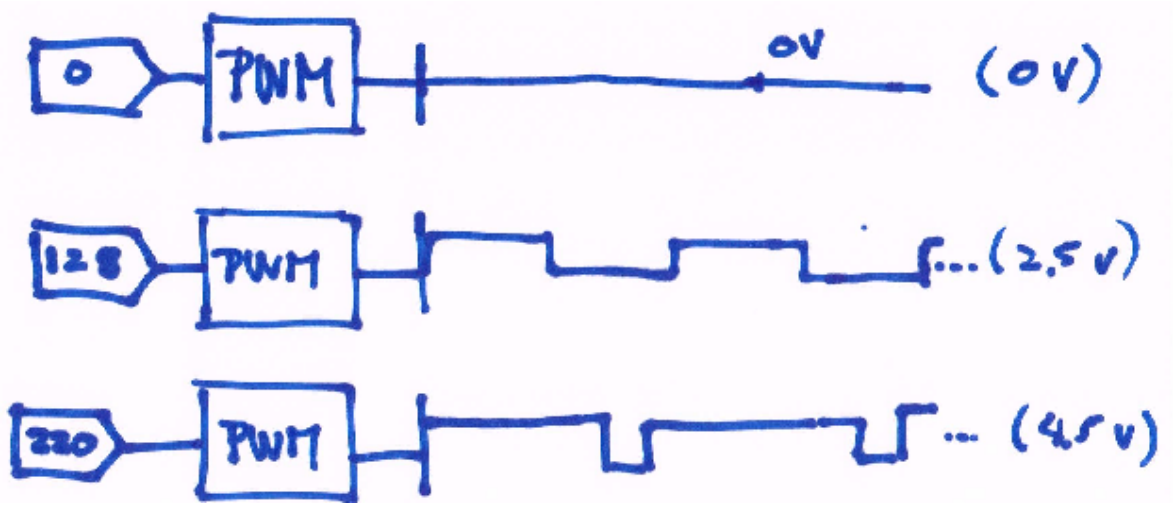
Entradas analógicas del Arduino

```
int dato; // variable que va a almacenar lo que lee un pin
int pin = 0; // pin A0 de entrada analogica

void setup(){
  // no es necesario inicializar nada
}

void loop(){
  dato = analogRead(pin);//dato es un entero entre 0 y 1023
}
```

Salidas analógicas (PWM) del Arduino



- Pueden *emular* 256 voltajes distintos entre 0 y 5V.
- Por ejemplo para controlar motores de un robot, intensidad de un LED
- Funcion `analogWrite(pin,valor)`
  - `pin` pin de salida del Arduino (debe tener ~)
  - `valor` número entre `0` (0V) y `255` (5V)

Ejercicios

Ejercicio 3

Determinar qué valor entero corresponde a una tensión de entrada de 3,5 volts en un pin de entrada analógica.

Ejercicio 4

Determinar el valor a poner en un pin de salida PWM para emular una tensión de 3,5 volts.

Ejercicio 1: LED con control de intensidad

Circuito

Armar un circuito en el protoboard que conecte la salida de un divisor resistivo a una entrada analógica del Arduino, y una salida PWM del Arduino a un LED.

Programa

Escribir, compilar y ejecutar un programa que repita indefinidamente esta secuencia:

- lea el valor del pin analógico conectado al d.r.
- convierta el valor del rango de entrada $0 - 1023$ al rango de salida $0 - 255$
- escriba el valor convertido al pin de salida conectado al LED

Arquitectura del sistema

Arquitectura del sistema

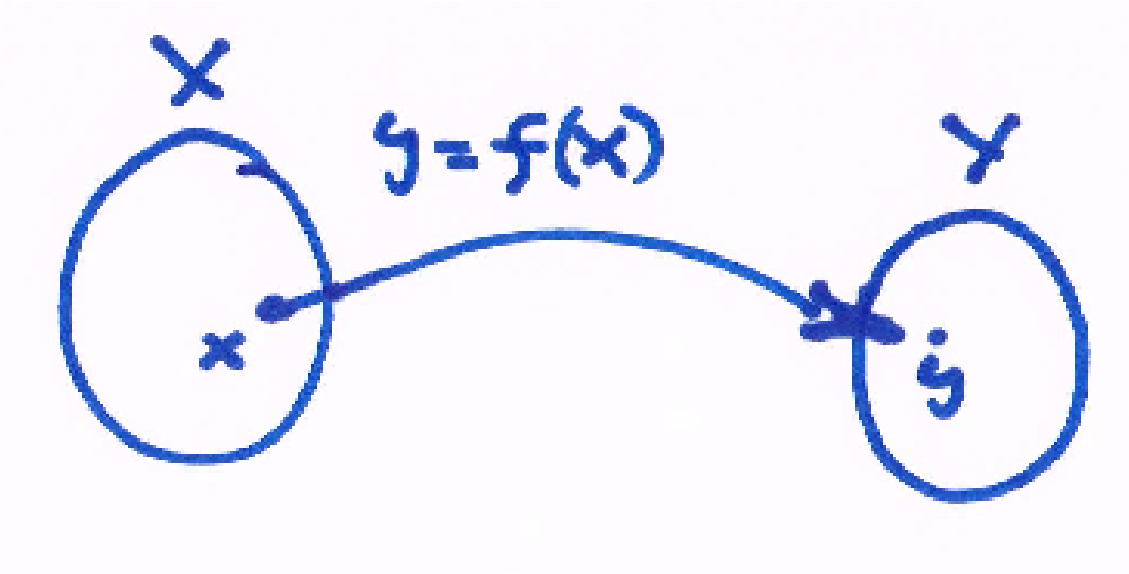
Von Neumann

- Memoria (estado interno)
- Programa
- Entrada de datos
- Salida de datos

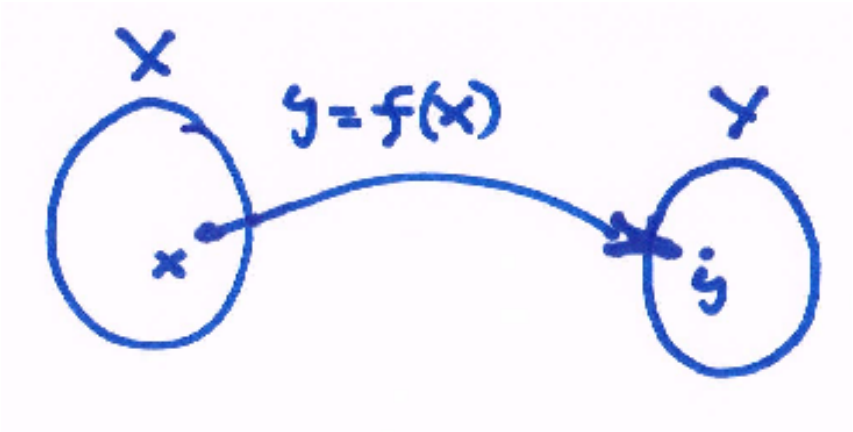
Programación estructurada en arduino

- Constantes
- Variables globales
- Tipos de variables
- Funciones y variables locales
- Dispositivos de E/S

Funciones



Funciones (concepto general)



- Abstracción matemática
- recibe un conjunto de parámetros como entrada . . .
- produce un resultado como salida

Funciones en Arduino

- Conjunto de sentencias en un bloque ({})
- Nombre único, conjunto de parámetros (opcional)
- Valor de retorno (opcional)

```
float promediar(float param1, float param2) {
  float p;
  p = (num1 + num2) / 2.0;
  return p;
}
```

Llamada a funciones

Definición

```
// DEFINIMOS promediar:
// recibe 2 parametros float
// devuelve un float
float promediar(float num1, float num2) { //encabezado
  float p;  // variable local
  p = (num1 + num2) / 2.0; // sentencia
  return p; // termina (RETORNA) y devuelve resultado
}
```

Llamada

```
void loop() {
  int k;
  // LLAMAMOS a la funcion 'promediar' con param1=4, param2=5
  // al RETORNAR el resultado se guarda en variable k
  k = promediar (4, 5);
}
```

Por qué usar funciones?

Sintetizar tareas repetitivas

- ahorra código,
- reduce errores

Encapsular

- *organiza* programa en conceptos y tareas
- facilita el desarrollo
- reutiliza código

Funciones y Arduino

API: Application Program Interface

- Definidas en Arduino, llamadas por el programador
- Ejemplos: `delay`, `analogWrite`, `digitalRead`, `pinMode`
- Hay muchas mas!

Callbacks – ciclo de vida

- Definidas por el programadar y llamadas por el Arduino
- `setup`: llamada al prenderse o resetearse el Arduino
- `loop`: llamada luego de setup, una y otra vez

Definidas por el usuario

- Pueden definirse todas las que se quiera dentro del Sketch

Bibliotecas

- Conjuntos de funciones hechas por otros para algo particular
- Ejemplo: manejo de motores, control remoto, wi-fi, etc.

Funciones y Arduino

API: Application Program Interface

- Definidas en Arduino, llamadas por el programador
- Ejemplos: `delay`, `analogWrite`, `digitalRead`, `pinMode`
- Hay muchas mas!

Callbacks – ciclo de vida

- Definidas por el programadar y llamadas por el Arduino
- `setup`: llamada al prenderse o resetearse el Arduino
- `loop`: llamada luego de setup, una y otra vez

Definidas por el usuario

- Pueden definirse todas las que se quiera dentro del Sketch

Bibliotecas

- Conjuntos de funciones hechas por otros para algo particular
- Ejemplo: manejo de motores, control remoto, wi-fi, etc.

Ejercicio 2: funciones

Entrada analógica a volts

Escribir una función `float convertirEntrada(int valorEntrada)` que reciba el entero leído de una entrada analógica y devuelva el en voltios correspondiente

volts a PWM

Escribir una función `int convertirSalida(float valorVolts)` que tome un valor en voltios de 0 a 5 y genere el valor de PWM (0 a 255) que genere ese voltaje en la salida.

Repetidor analógico

Reescribir el orograma del Ejercicio 1 usando las funciones `convertirEntrada` y `convertirSalida`. Ejecutar y probar.

Ejercicio 3: Repetidor de intervalos

Idea

Se trata de armar un circuito que mida el intevalo de tiempo entre dos pulsaciones de un botón y luego utilice ese valor como período para encender y apagar un led indefinidamente.

Circuito

Conectar un *pulsador* a una entrada digital del Arduino y un *LED* (en serie con una resistencia) a una salida digital del Arduino.

Ejercicio 3: Repetidor de intervalos

Programa

Memoria: variables `contador` e `intervalo`
setup configurar los pines de la manera descrita
loop debe hacer lo siguiente:

1. Medir el pin de entrada cada 100ms hasta que valga 1
2. Almacenar el tiempo del sistema en la variable `contador`
3. Medir el pin de entrada cada 100ms hasta que valga 0
4. Medir el pin de entrada cada 100ms hasta que valga 1
5. Almacenar la diferencia entre el tiempo del sistema y el `contador` en la variable `intervalo`
6. Repetir indefinidamente:
   - Encender el LED durante 100ms
   - Apagar el LED durante (`intervalo`-100) ms

Ejercicio 4: Voltímetro de LEDs

- Montar un circuito que maneje cuatro LEDs en fila y tome la entrada de un *divisor resistivo* colocado entre GND y VCC (ver figura)
- programar al Arduino para que cada 50ms lea una entrada analógica y luego prenda o apague los cuatro LEDs de manera que:
  - el primer pin se prenda si el voltaje en la entrada supera $1{,}0V$
  - el segundo pin se prenda si el voltaje en la entrada supera $2{,}0V$
  - el tercer pin se prenda si el voltaje en la entrada supera $3{,}0V$
  - el cuarto pin se prenda si el voltaje en la entrada supera $4{,}0V$