UPMDIE
INDUSTRIALES

4 – Dispositivos de E/S

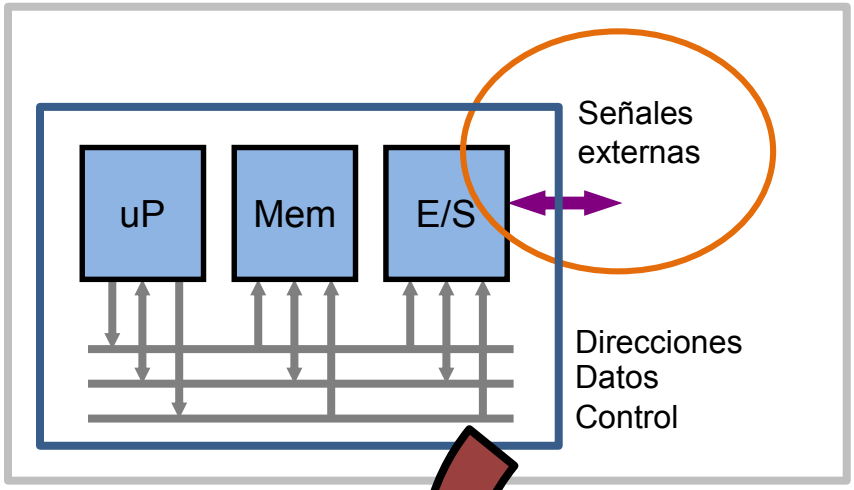
- *Dispositivos, puertos y registros de E/S*
- *Dispositivos de E/S paralelo*
- *Conversión A/D. Pulse Width Modulation*
- *Otros dispositivos típicos de E/S*

cei@upm.es

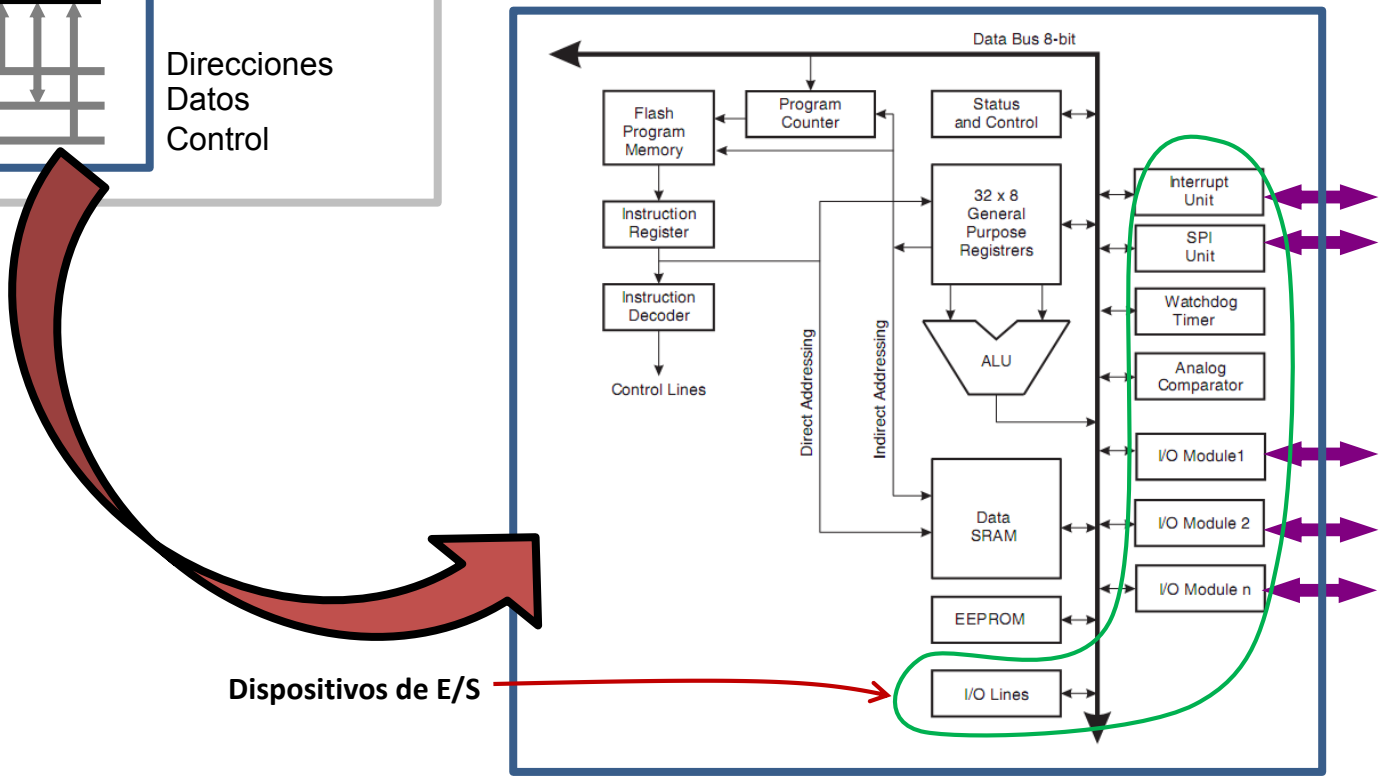
POLITÉCNICA

Dispositivos de Entrada/Salida

- Los dispositivos de E/S permiten leer y escribir las señales externas



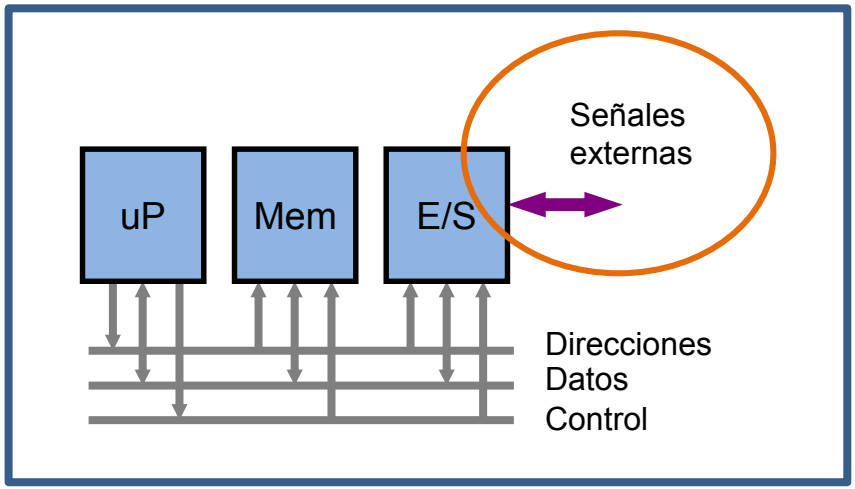
En el caso de la familia AVR de Atmel, la funcionalidad y el número de dispositivos de E/S depende del modelo de MCU en particular (la CPU es la misma para toda la familia AVR)



Puertos y registros de Entrada/Salida

Puerto de E/S

- Conjunto de señales externas con una funcionalidad común y una circuitería (dispositivo de E/S) que permite leerlas y/o controlarlas.
- Diferentes tipos: E/S paralelo, E/S serie, E/S analógica, ...
- Los puertos de E/S pueden tener asociadas diferentes funcionalidades (según se configuren)



Registros de E/S

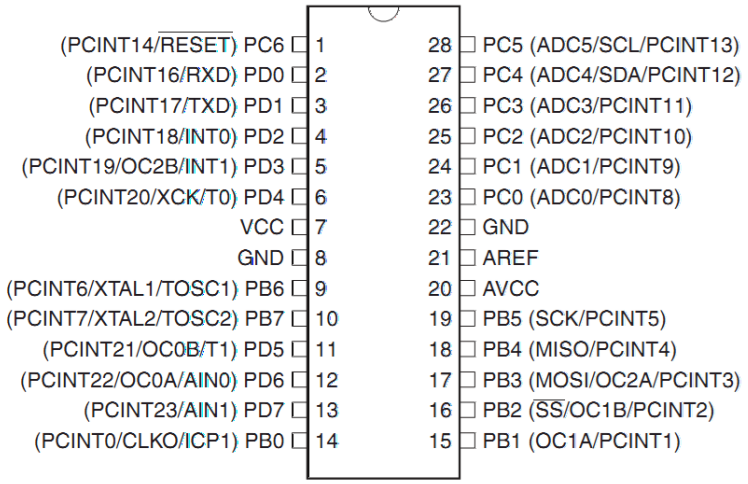
- Registros asociados a los puertos de E/S. Cada puerto de E/S puede tener asociados uno o varios registros que permiten controlar el funcionamiento del puerto.
- Los registros de E/S se acceden mediante instrucciones de lectura/escritura de forma análoga a como se accede a las posiciones de memoria.
- Existen registros asociados a los datos (las señales a controlar) y al control (el modo de operación).
- El tamaño (nº de bits) del registro suele coincidir con el de la palabra del uP
- Normalmente interesa más el valor particular de cada bit de una palabra, y no tanto el valor de la palabra completa

Ejemplo: Puertos de E/S en el AtMega168

- Los puertos de E/S en los AVR pueden operar en varios modos (como en casi todos los uC)

Puertos de E/S

- **Puerto B (PB)**
  - Entrada/Salida paralelo (se verá)
  - Generación de señales PWM (se verá)
  - Otros (SPI, interrupciones, capturas..)
- **Puerto C (PC)**
  - Entrada/Salida paralelo
  - Convertidor A/D (se verá)
  - Otros (Interrupciones)
- **Puerto D (PD)**
  - Entrada/Salida paralelo
  - Puerto serie
  - Generación de señales PWM
  - Otros (Interrupciones)



Puertos de E/S paralelo

Puertos de E/S paralelo

- Puerto de E/S en el que se controlan (leen o escriben) varias señales externas al mismo tiempo.
- Pueden ser de entrada, salida, o configurables por el programa.
- Son los puertos más comunes.
- El tamaño (nº de bits) del puerto suele coincidir con el de la palabra del uP (8 bits, en el AVR)

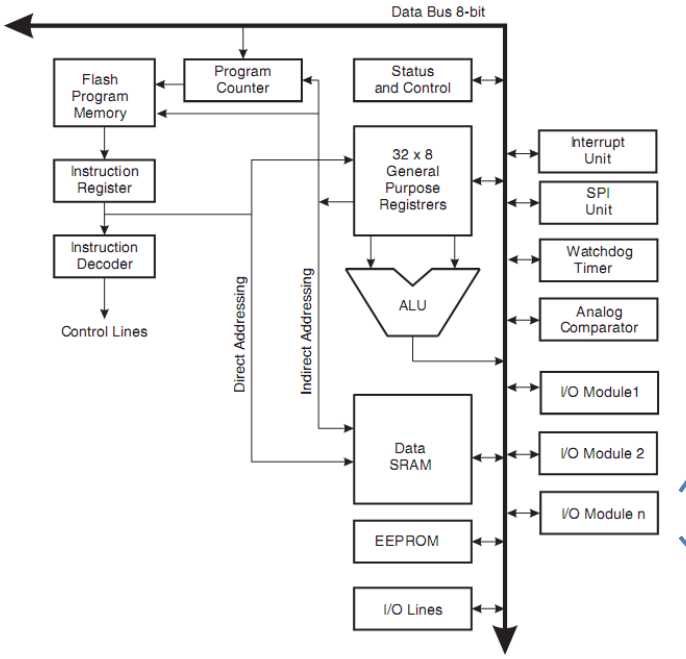
Registros asociados

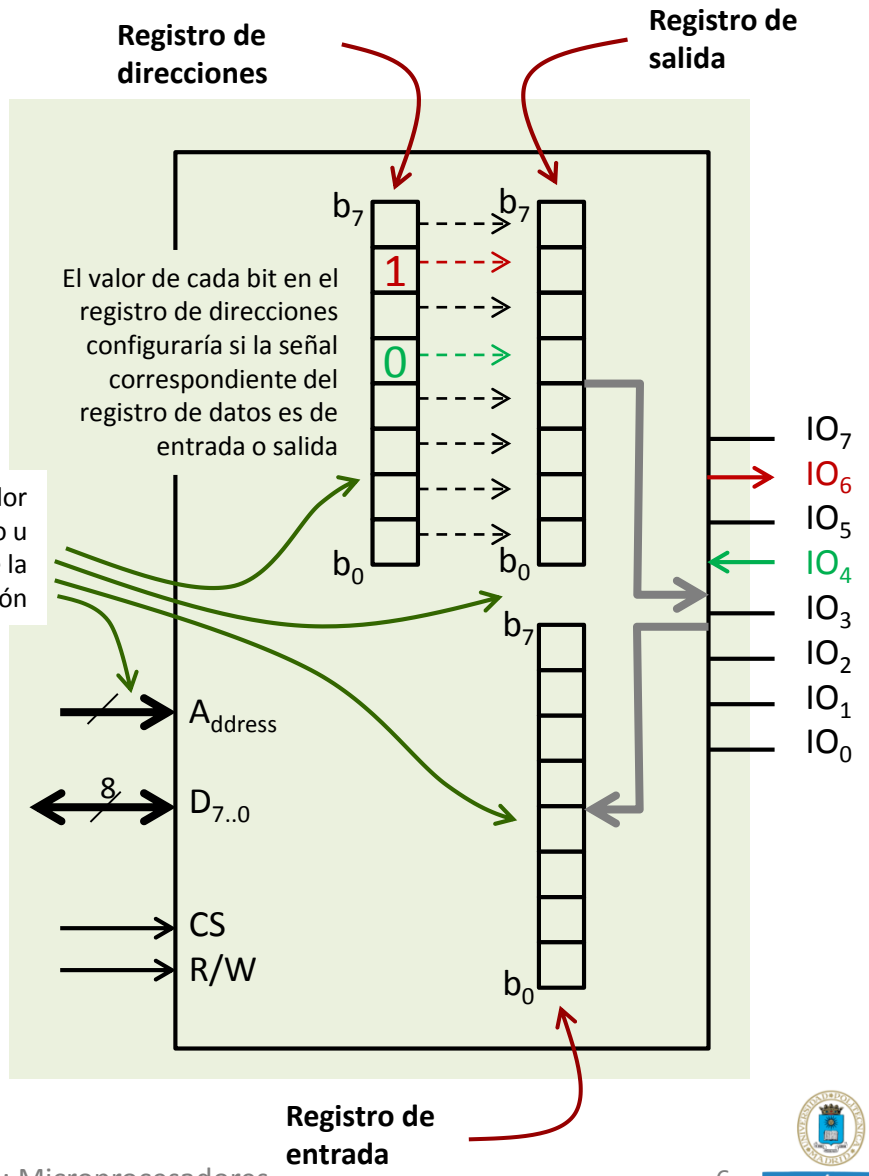
- **Registros de datos**: registro conectado a las señales de E/S.
  - Si un bit del puerto está configurado como entrada un nivel te tensión alto (Vcc) en su pin correspondiente se leerá como un '1', y un nivel bajo (Gnd) se leerá como un '0'.
  - Si un bit está configurado como salida, un '1' fijará un valor alto (Vcc) en su pin correspondiente, y un '0' pondrá un valor bajo (Gnd).
  - Escribir sobre un bit configurado como entrada no tiene ningún efecto.
  - En algunos uC, los registros de datos de entrada (para leer) y de salida (para escribir), son distintos (como en el caso de los AVR que usamos)

- **Registro de direcciones de los datos**: registro que fija la dirección de las señales de E/S.
  - Depende de cada micro, pero es habitual que un '0' provoque que su señal correspondiente sea de entrada y un '1' haga que sea de salida.

Puertos de E/S paralelo en el AtMega168

- Dispone de tres puertos de E/S paralelo (B, C y D)

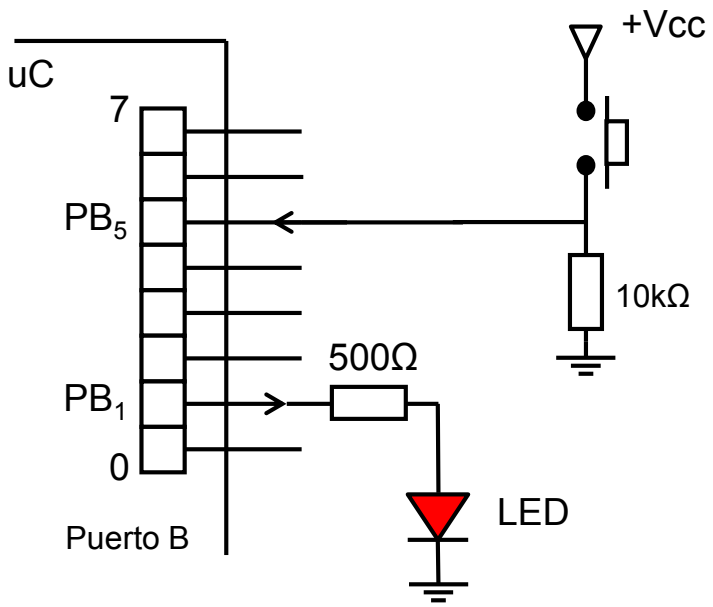




- **Registro de entrada**: registro de lectura de puerto
  - Un bit se lee como un '1' si hay un '1' (Vcc) en su pin correspondiente
  - Un bit se lee como un '0' si hay un '0' (Gnd) en su pin correspondiente
- **Registro de direcciones** (Data Direction Register):
  - Configuración del sentido de cada bit del puerto ('0' lo pone como entrada y '1' como salida)
- **Registro de salida:** Registro de escritura.
  - Al escribir un '1' en un bit se pone un '1' (Vcc) en su pin correspondiente
  - Al escribir un '0' en un bit se pone un '0' (Gnd) en su pin correspondiente

Ejemplo: Encender y apagar un LED con un pulsador

- ¿Cómo podríamos hacer un sistema que encienda las luces cuando anochece?



2

```
unsigned char boton;
unsigned char ledState = 0;

// Primero planteo la funcionalidad, luego ya veremos
// cómo accedo al hardware
int main() {
  // Inicialización
  InicializaES();

  while (true) { // Bucle infinito
    boton = leerBoton();      // leo el estado del botón
    if (boton != 0) { // boton pulsado → actuar
      if (ledState == 0) {
        setLed(1);
        ledState = 1;
      } else {
        setLed(0);
        ledState = 0;
      }
    }
  }
}
```

1

Planteamiento de alto nivel

a) Configurar las señales de entrada y salida
b) Leer el bit del puerto que me interesa
c) Decidir en función del valor
d) Escribir en el puerto actuando sólo sobre el led
e) Repetir desde el paso b

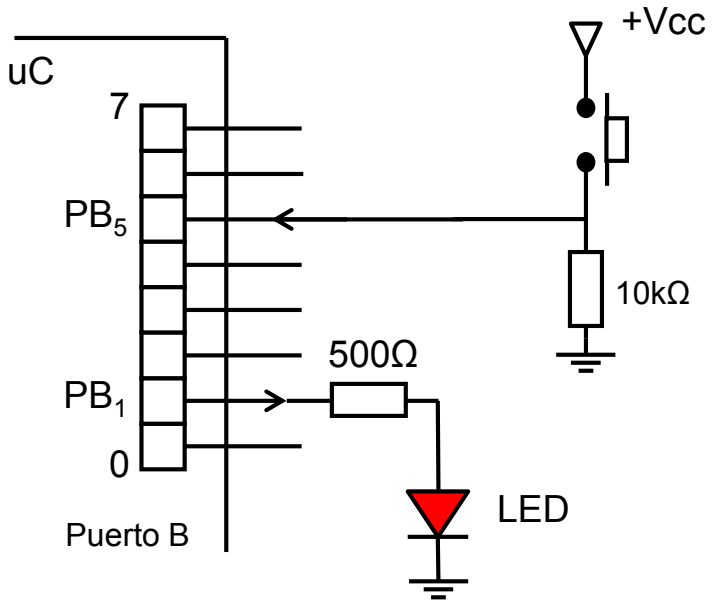
3

⁇ Si comprobásemos el funcionamiento del sistema, se observaría que mientras se mantiene el botón pulsado el led luciría levemente. Por otro lado, veríamos que el led se enciende y apaga de forma poco predecible en vez de conmutar como queremos (debería cambiar de estado cada vez que pulsamos)

→¿A qué podría ser debido este comportamiento?
→¿Cómo podría solucionarse?

Ejemplo: Encender y apagar un LED con un pulsador

- Implementando las funciones de E/S, (acceso al hardware)

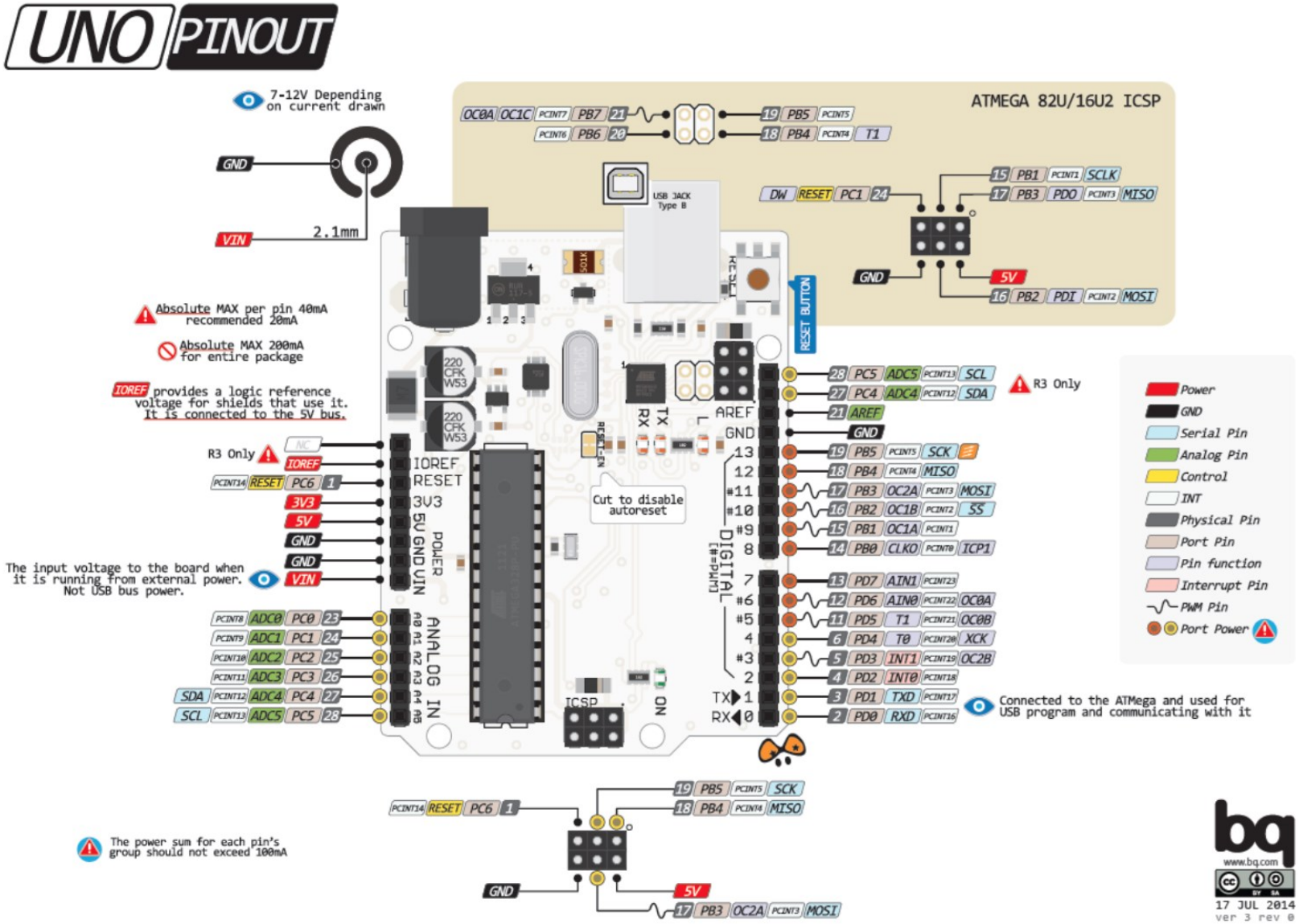



```
unsigned char boton;
unsigned char ledState = 0;

// Primero planteo la funcionalidad, luego ya veremos
// cómo accedo al hardware
int main() {
  // Inicialización
  InicializaES();

  while (true) { // Bucle infinito
    boton = leerBoton();      // leo el estado del botón
    if (boton != 0) { // boton pulsado → actuar
      if (ledState == 0) {
        setLed(1);
        ledState = 1;
      } else {
        setLed(0);
        ledState = 0;
      }
    }
  }
}
```

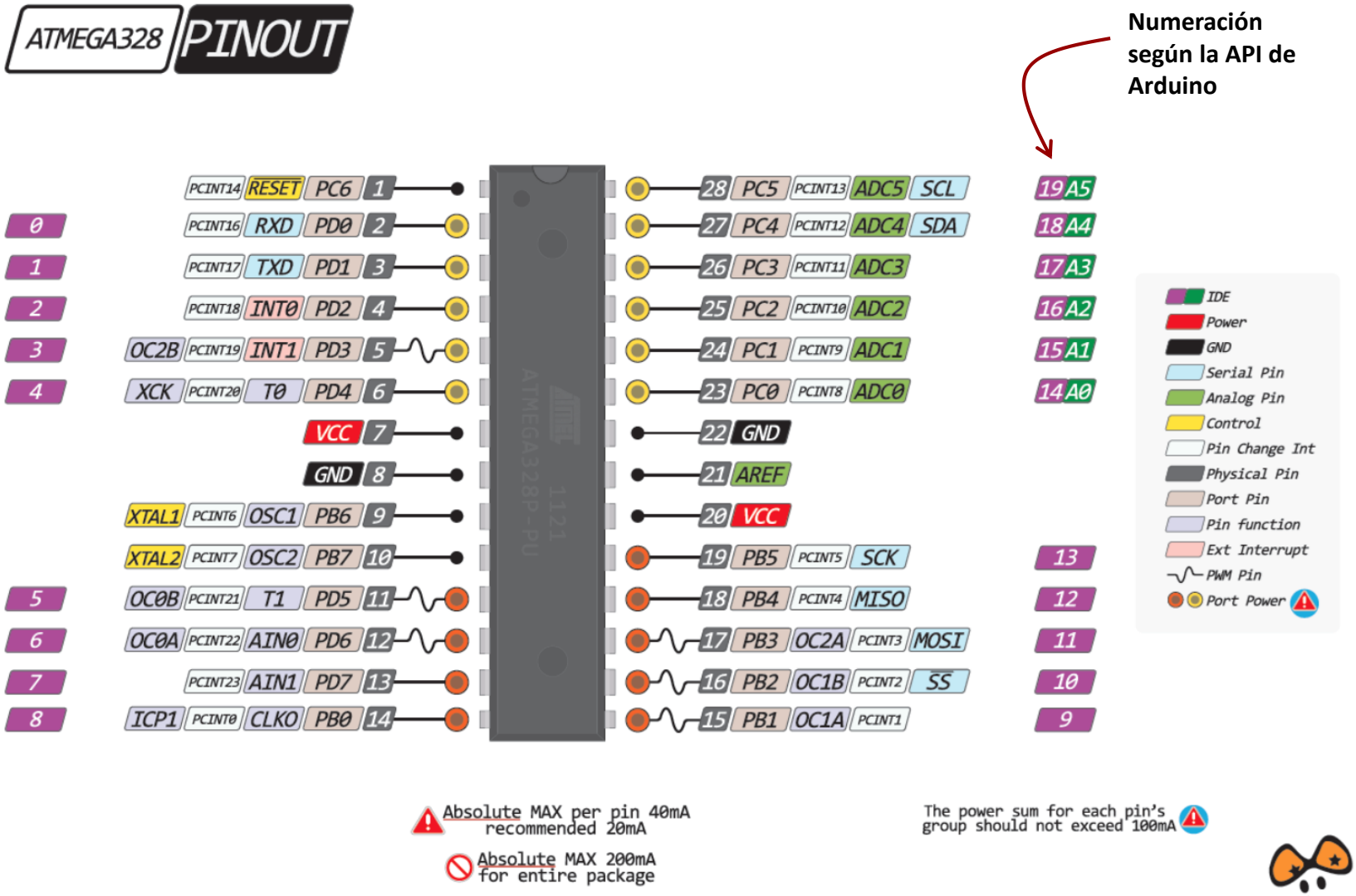
```
void InicializaES() {
  // Configura el pin 1 el puerto B para que sea salida
}

char leerBoton() {
  // lee el valor del pin 5 del puerto B
}
```

```
void setLed(char state) {
  // Modifica el valor del pin 1 del puerto B
  // en función de 'state'
}
```

Ejemplo: Encender y apagar un LED con un pulsador

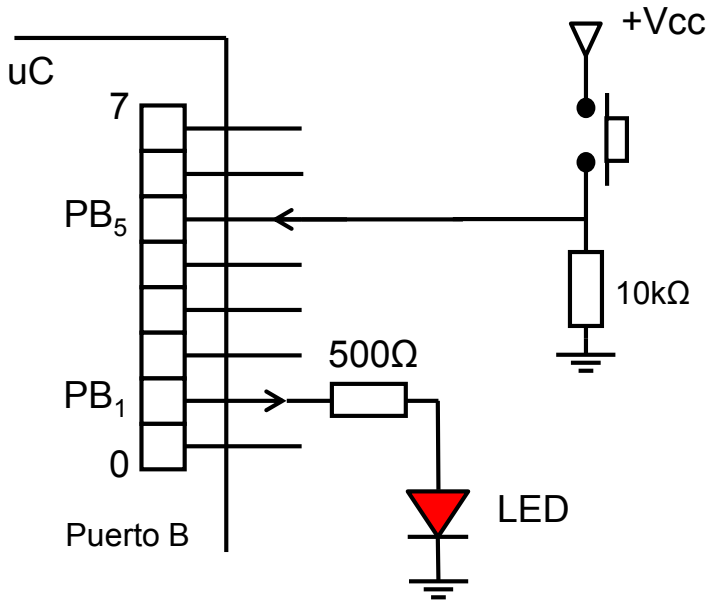


Ejemplo: Encender y apagar un LED con un pulsador



Ejemplo: Encender y apagar un LED con un pulsador

- Lo mismo con la **API** de Arduino



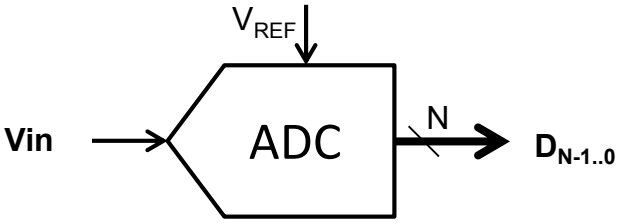

```
unsigned char boton;
unsigned char ledState = 0;

void setup() { // esta función sólo se ejecuta una vez
  pinMode(  9, OUTPUT ); // Configura los pines
  pinMode( 13, INPUT );
}

void loop() { // Esta función se ejecuta continuamente
  boton = digitalRead(13);
  if (boton != 0) { // boton pulsado → actuar
      if (ledState == 0) {
        digitalWrite(9, HIGH);
        ledState = 1;
      } else {
        digitalWrite(9, LOW);
        ledState = 0;
      }
    }
}

////////////////////// esto no lo haríamos nosotros, ya que
int main() {       // el entorno de arduino incluye la
  setup();         // función main() automáticamente, así
  while (true) {   // que nosotros solo tenemos
    loop();        // que hacer las funciones de setup(),
  }                // que se ejecuta solo una vez, y loop()
}                  // que se ejecuta repetidamente
```

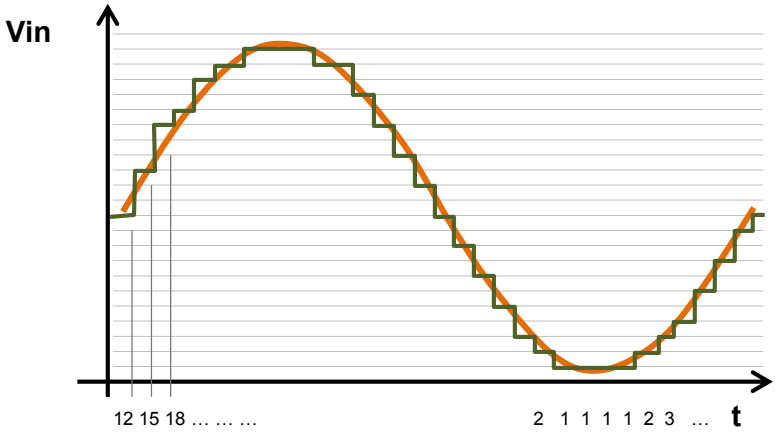
Nota: En la placa de Arduino, el pin 9 de la placa se corresponde con el pin PB1 del AtMega, y el pin 13 de la placa se corresponde con el pin PB5 del AtMega

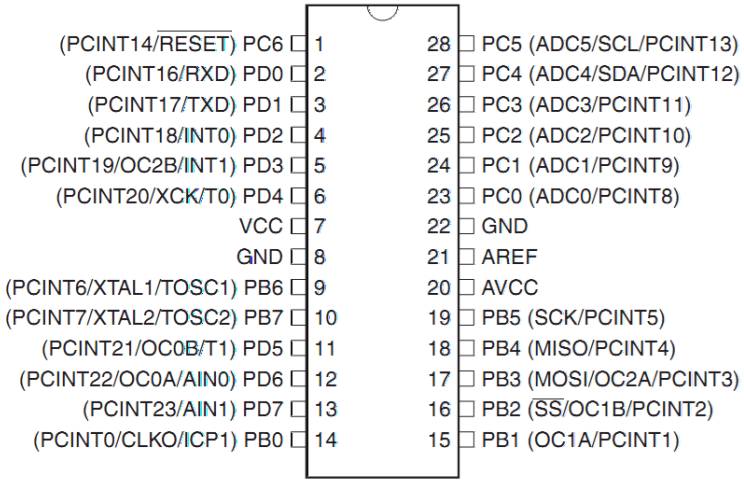
API (Application Programming Interface): es un conjunto de funciones y procedimientos que ofrece una biblioteca como capa de abstracción para acceder a funcionalidades de un nivel inferior.

Convertidor Analógico-Digital (ADC)

- Digitaliza una señal continua (analógica) en 2^N niveles.







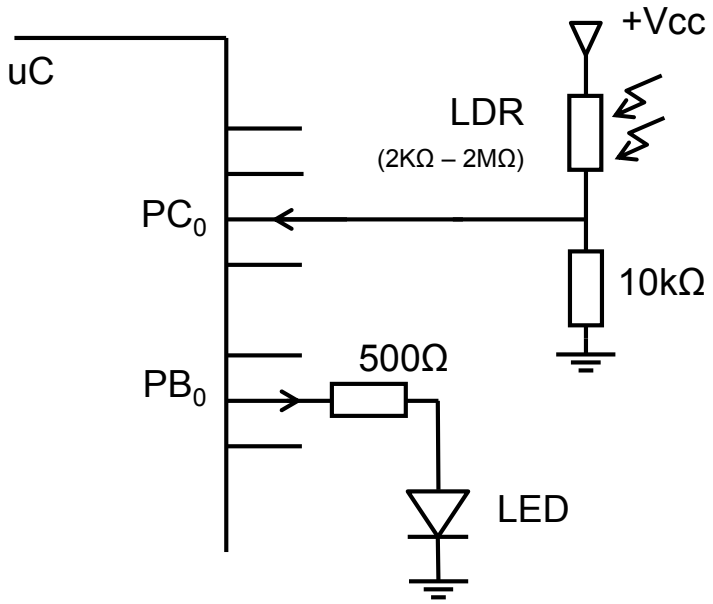
- Características generales:
  - Número de bits del convertidor (N) → resolución
  - Linealidad
  - otras ...

Resolución $q = \frac{V_{REF}}{2^N}$

Ej: $V_{REF} = 5V$, $N = 10$ ➡ $q = 4{,}88$ mV

Ejemplo: Controlar un LED con la luz ambiente

- ¿Cómo podríamos hacer un sistema que encienda las luces cuando anochece?




```
...
int analogVal;
unsigned char led = 0;
...
configurar PB0 como salida (*);
configurar ADC para leer el valor analógico en PC0 (*);
...

while (true) { // bucle infinito
  analogVal = valor devuelto por el ADC (0 – 1023) (*);
  if (analogVal < 100) { // 100 sería un valor obtenido
                         // al anochecer, experimentalmente
    encender led (*);
  } else {
    apagar led (*);
  }
}

... (*) se verá luego cómo se haría usando Arduino
```

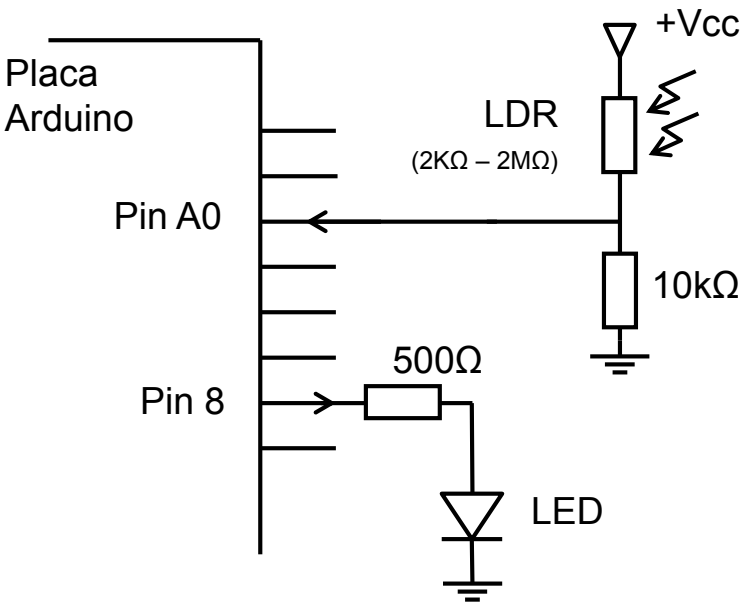
1. Configurar el convertidor A/D y la señal de salida
2. Leer el valor del convertidor A/D
3. Decidir en función del valor
4. Actuar sólo sobre el led
5. Repetir desde el paso 2

⁇ Al comprobar el funcionamiento del sistema, se observa que al anochecer y al amanecer la luz vibra (se enciende y apaga de forma irregular) durante un rato.
→¿A qué puede ser debido ese comportamiento?
→¿Cómo podría solucionarse?

⁇ También se observa que, a veces, al pasar alguien por delante del sensor se enciende la luz y aun no ha anochecido
→ ¿Cómo podría solucionarse?

Ejemplo: Controlar un LED con la luz ambiente

- ¿Cómo podríamos hacer un sistema que encienda las luces cuando anochece utilizando la placa y la API* de Arduino?

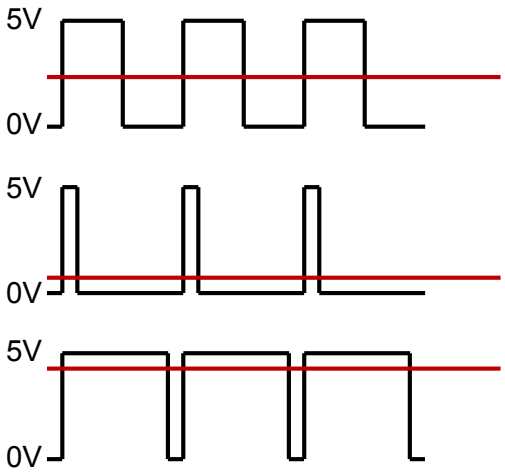


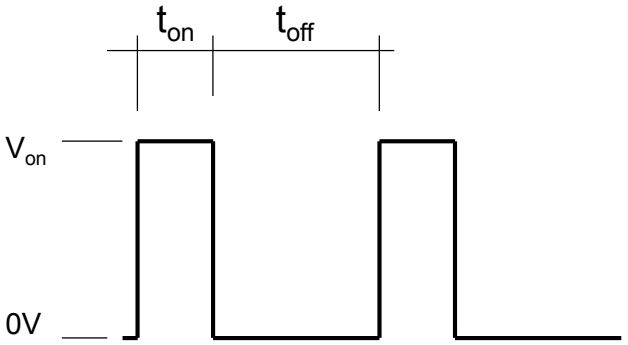

```
void setup() { // esta función sólo se ejecuta una vez
  pinMode( 8, OUTPUT ); // Configura los pines
  pinMode( A0, INPUT ); // Puede no hacer falta, ya que
                        // tras el reset las E/S suelen
                        // estar como entradas
}

void loop() { // Esta función se ejecuta continuamente

  int analogVal = analogRead( A0 ); // valores de 0 a 1023

  if (analogVal < 100) { // 100 sería un valor obtenido
                         // al anochecer experimentalmente
    digitalWrite( 8, HIGH );
  } else {
    digitalWrite( 8, LOW );
  }
}
```

Nota: En la placa de Arduino, el pin A0 de la placa se corresponde con el pin PC0 del AtMega, y el pin 8 de la placa se corresponde con el pin PB0 del AtMega

Modulación por ancho de pulso (PWM)

- Para generar una señal «analógica», la mayoría de los uC no incluyen un convertidor digital-analógico (DAC), sino que utilizan el valor medio de una señal digital periódica con ancho de pulso variable

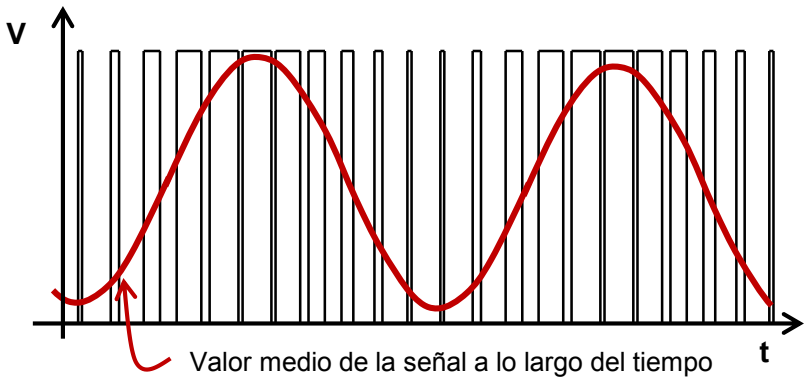




$V_{medio} = V_{on} * t_{on} / (t_{on} + t_{off}) = V_{on} * d$

d (duty cycle) = $t_{on} / (t_{on} + t_{off})$

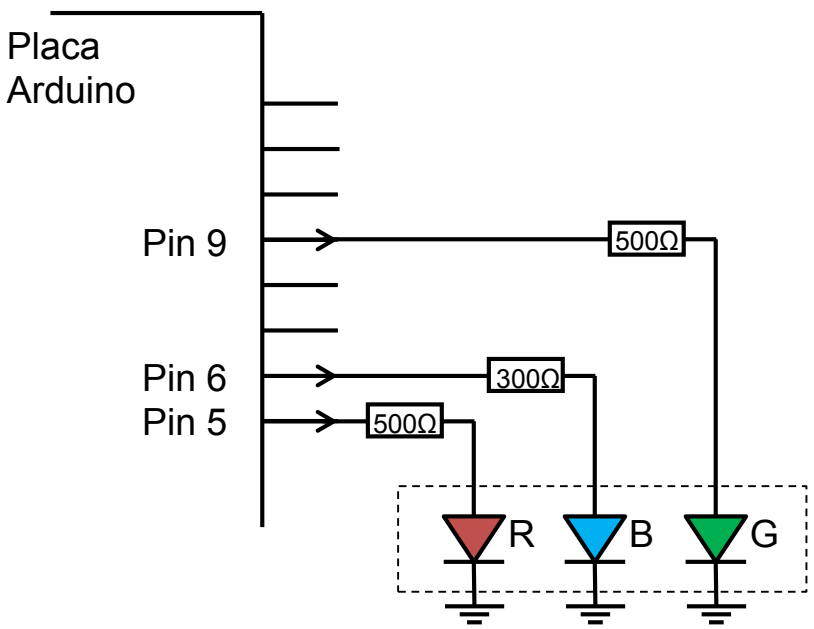
% del tiempo que la señal esta encendida



- Se usa muy a menudo para el control de motores y luces, o en generación de audio (o a veces también como método de comunicación, p.ej: radiocontrol).
- La frecuencia de la señal ($1/(t_{on}+t_{off})$) debe ser suficientemente alta como para que el sistema controlado responda a su valor medio, y no al instantáneo.

Ejemplo: Controlar un LED RGB

- Un led RGB está compuesto de tres leds de colores rojo (R), verde (G) y azul (B). El color final depende de la intensidad de cada uno




```
#define LED_R 5
#define LED_G 9
#define LED_B 6

void setup() {
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);
}

void setColor(int r, int g, int b) {
  analogWrite(LED_R, r); // Valores de 0 a 255
  analogWrite(LED_G, g);
  analogWrite(LED_B, b);
}

// cambia el color de forma progresiva (solo la
// componente del rojo)
int inc = 1;
int r = 0;
int loop() {
  setColor(r, 200, 100);
  r = r + inc;
  if (r == 255) inc = -1;
  if (r == 0)   inc =  1;
  delay(10); // paramos un poco (si no, no lo vemos)
}
```

Nota: Con la API y la placa de Arduino, sólo es posible controlar 6 señales PWM (en concreto, los pines 3, 5, 6, 9, 10 y 11 de la placa), y su frecuencia es fija (490Hz aprox)

Otros dispositivos de E/S típicos

Tipo	Sirven para	Ejemplos (estándares)
E/S serie	Comunicaciones	UART, USART, USB, I2C, FireWire, CAN, etc.
Temporizadores y contadores	Control (en general) Datado (en particular)	
Perro guardián (watchdog)	Control de la ejecución correcta de un programa	
Controladores específicos	Discos duros, disquetes, etc.	SCSI, IDE, SATA, VGA, ...