Taller Arduino

Arquitectura de computadoras

Facultad de Ingeniería

Instituto de Computación

mina
network management | artificial intelligence

Temas

- Introducción a los sistemas embebidos
- Resistencias, LEDs y protoboards
- Microcontroladores y placas de entrada salida
- Interactuando con E/S

¿Qué es un Sistema Embebido?

- Un sistema embebido (SE) es un sistema computador destinado a una aplicación en particular.
- Los sistemas computadores de propósito general tienen muchas aplicaciones, según el software que se instale.
- Es una combinación de hardware, software y posibles elementos mecánicos.
- Específicos para una tarea por lo que son optimizados para la misma.

Firmware

- Rutinas de software almacenadas en memoria no volátil (Flash, ROM, EEPROM, etc).
- Software que se encuentra inmerso en el dispositivo de hardware a controlar.
- Es software muy acoplado con un hardware particular.

Características de los SE

- Interactúan con el entorno
  - Directamente sensando y controlando señales.
  - Comunicándose con otros dispositivos.
- Interacción con restricciones de tiempo real.
- Bajo consumo.

Entrada/Salida (E/S)

- Debido a su característica los sistemas embebidos deben interactuar con el ambiente que los rodea.
- Sensando señales del ambiente o actuando sobre el mismo.
- Hay dos maneras de manejar la E/S:
  - Digital
  - Analógica

E/S Digital

- Se intercambian “unos” y “ceros”.
- Esos valores corresponden a voltajes de referencia.
- Útil para controlar algunos dispositivos electrónicos:
  - Prender un led.
  - Leer estado de un botón.
- Utilizado para implementar protocolos de comunicación.

E/S Analógica(1/2)

- Los valores que se intercambian pueden tomar varios valores.
- Es necesario disponer de conversores Digital -> Analógico (D/A) o Analógico -> Digital (A/D).
- Un conversor A/D convierte un voltaje en una entrada a un valor digital.
- Un conversor D/A convierte un valor digital a un voltaje en un pin de salida.

E/S Analógica(2/2)

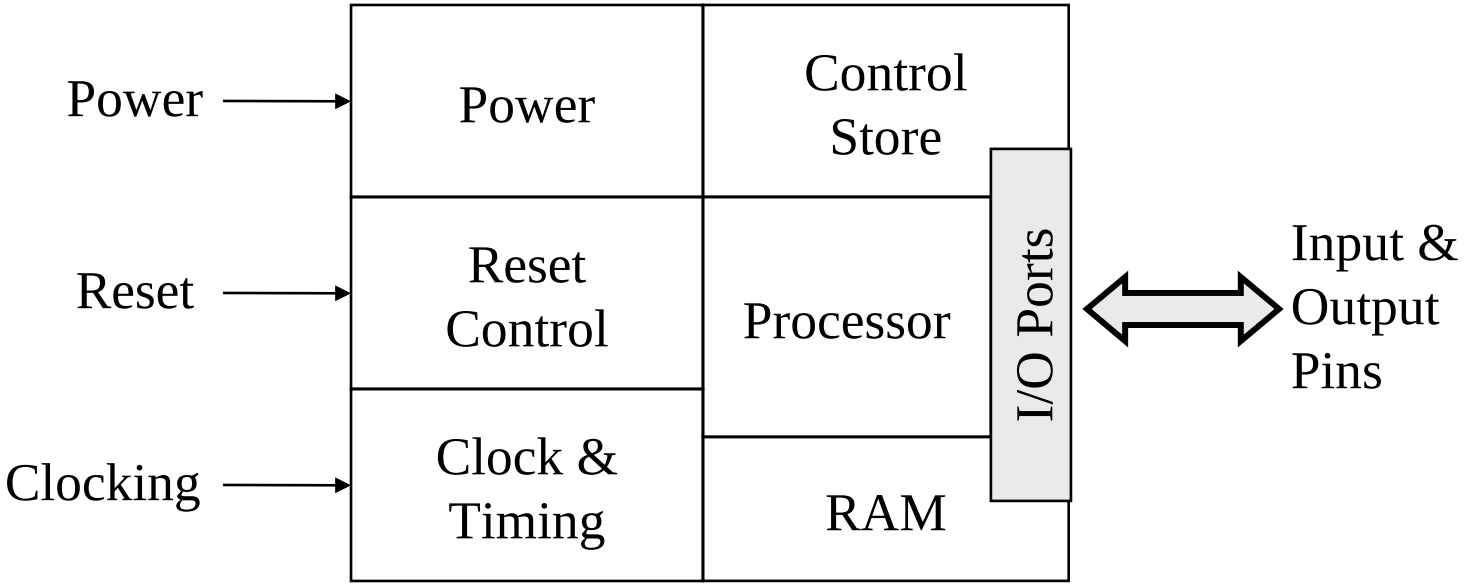
- La resolución del conversor condiciona la cantidad de valores a representar. Un conversor de 10 bits va a permitir representar valores desde 0 a 1023.
- Muchos sensores se manejan de esta manera: Temperatura, humedad, luz, micrófono
- Y también los actuadores: Parlantes, motores

Manejo de la E/S

- Paradigmas para implementar la lectura/escritura de E/S:
  - Polling: Donde se utilizan ciclos de CPU para estar constantemente consultando el valor de alguna entrada.
  - Interrupciones: El CPU es notificado externamente. A nivel de software se ejecuta una rutina de atención a la interrupción.

Microcontroladores

Un microcontrolador (µC) es un sistema autocontenido donde el microprocesador, soporte, memoria y entrada/salida se presentan dentro de un mismo integrado.



Registros de E/S

- Son los componentes más utilizados de los microcontroladores.
- Los microcontroladores disponen de registros para controlar los dispositivos de E/S.
- Espacios
  - E/S mapeada en memoria.
  - Mapa E/S y mapa de memoria.

Placas de entrada / salida

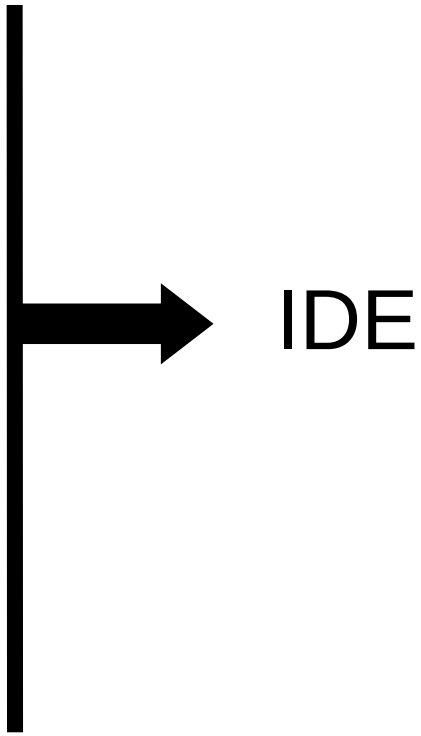
- Existen placas orientadas a trabajar con entrada/salida que permiten realizar diseños embebidos en base a ellas.
- Utilizadas para prototipar o diseñar productos de pocas unidades.
- Incluyen un microcontrolador, señal de reloj, conectores de E/S, memoria flash externa, acondicionamiento de señales.

Desarrollo de software

- Herramientas y entornos de desarrollo
- Programación
- Debug

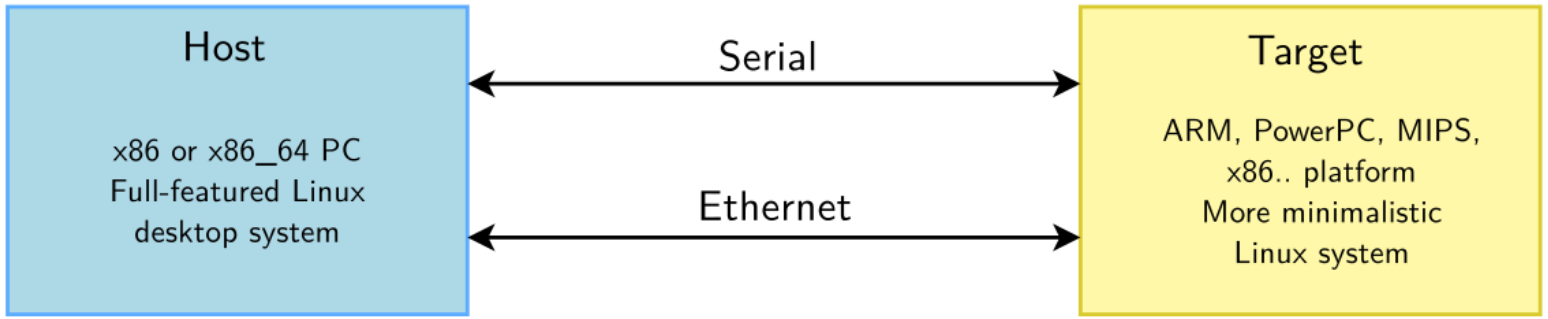
Herramientas y entornos de desarrollo (1/4)

- Editor
- Compilador
- Ensamblador
- Simulador
- Emulador
- Programador



Host vs. target

- Cross-compilation toolchain
- Cuando se trabaja en sistemas embebidos hay una clara división entre máquinas
  - Host
  - Target
- Estás máquinas deben estar conectadas de alguna forma.

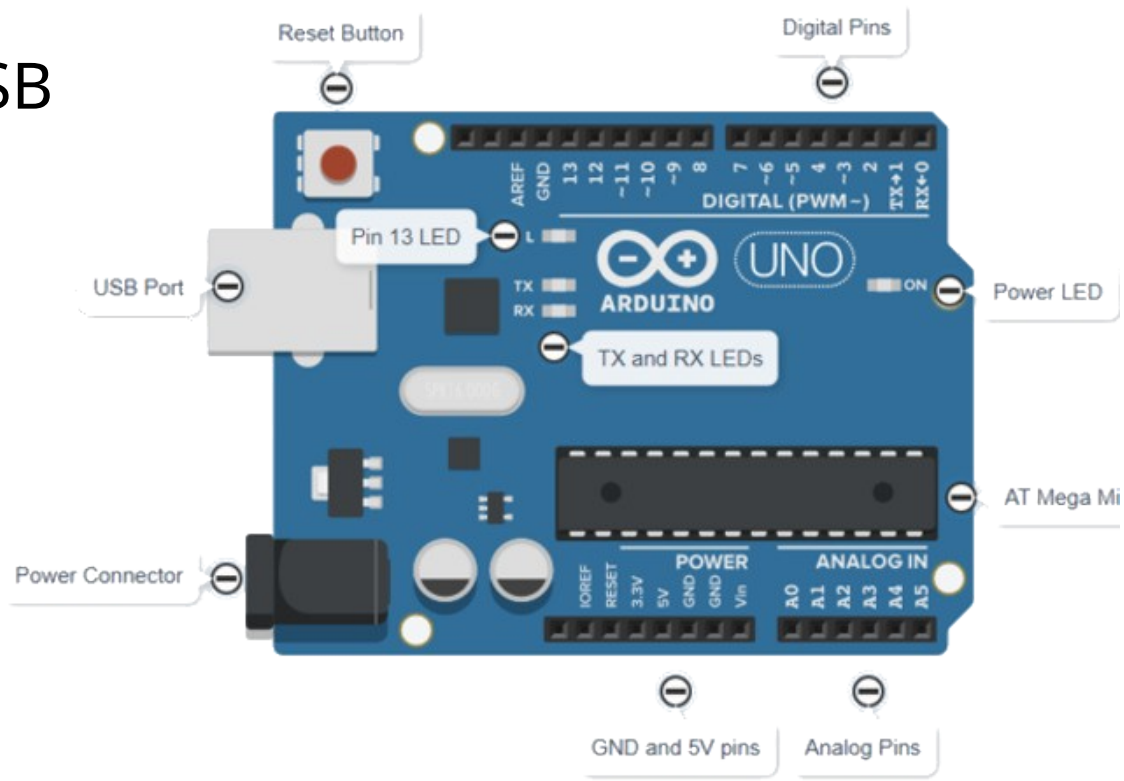


Arduino Uno

- Plataforma electrónica de hardware y software libre
- Utiliza un microcontrolador re-programable
- Tiene pines analógicos y digitales (entrada/salida)

Arduino Uno

- Alimentación:
  - A través de USB
  - 5 V
  - 250 mA
- Memoria
  - 32kB Flash
  - 2kB SRAM
  - 1kB EEPROM

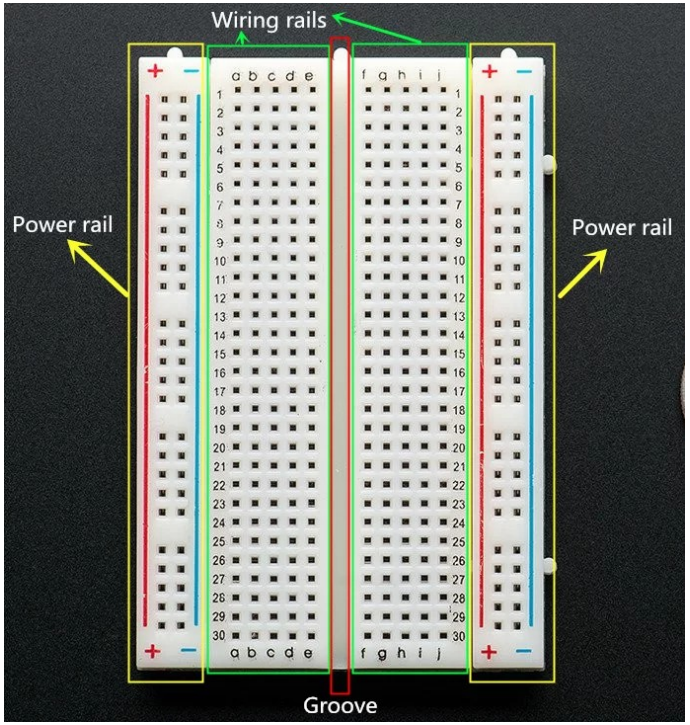


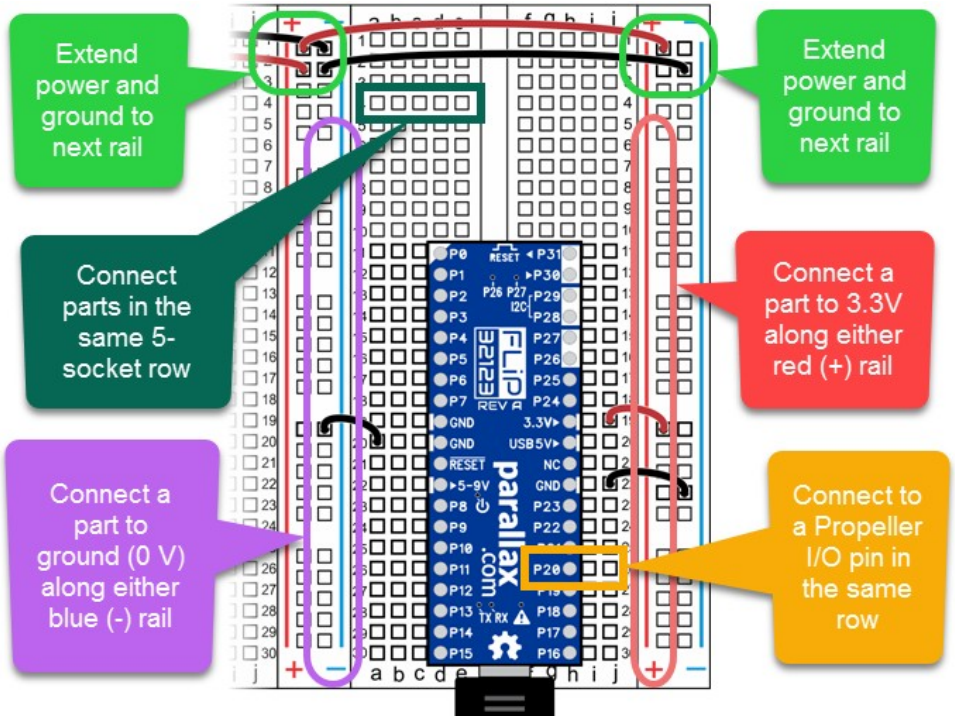
Cuidados :: Cosas a no hacer

- Cortocircuitar un pin de E/S con GND
- Cortocircuitar dos pines de E/S
- Aplicar sobretensiones en los pines de E/S
- Cambiar la polaridad de alimentación Vin y GND
- Cortocircuitar Vin y GND
- Aplicar tensión a los pines de salida de tensión

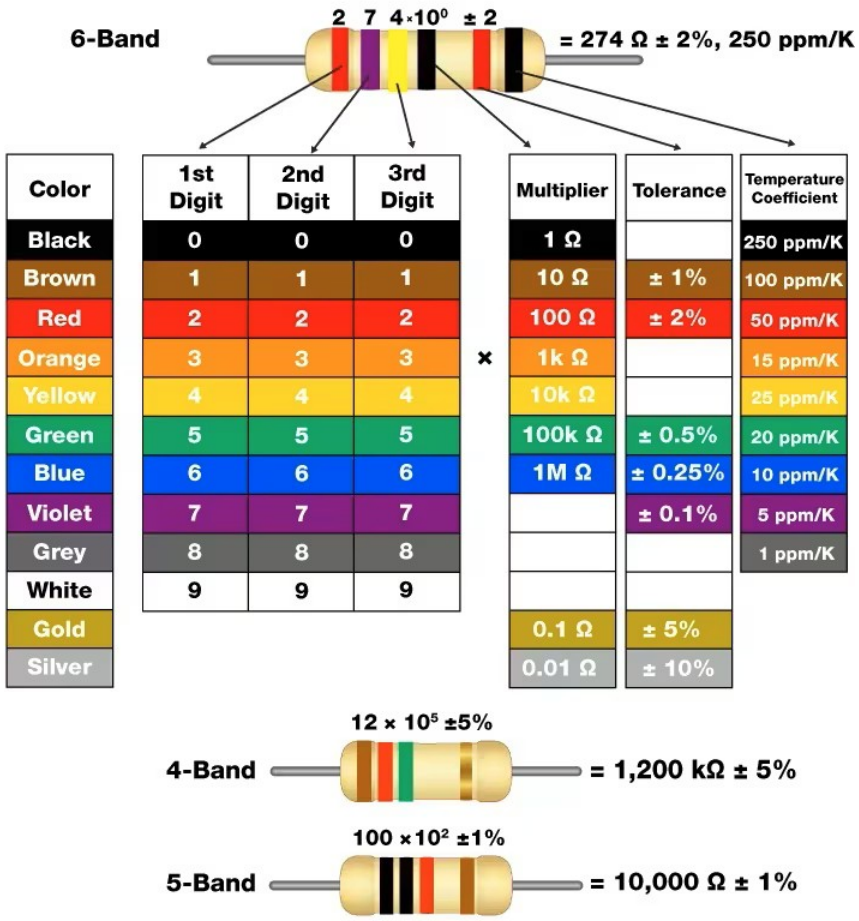
Protoboard

- Permite realizar circuitos de forma rápida, sin necesidad de soldar.

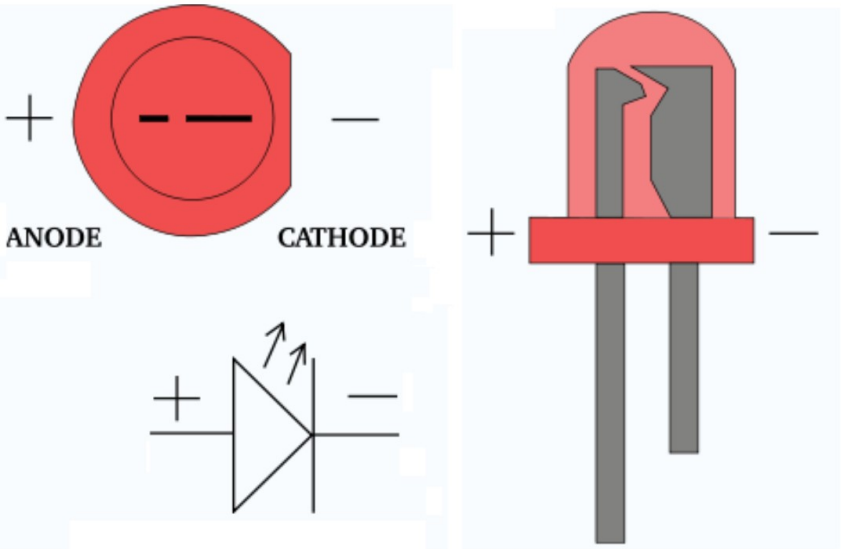


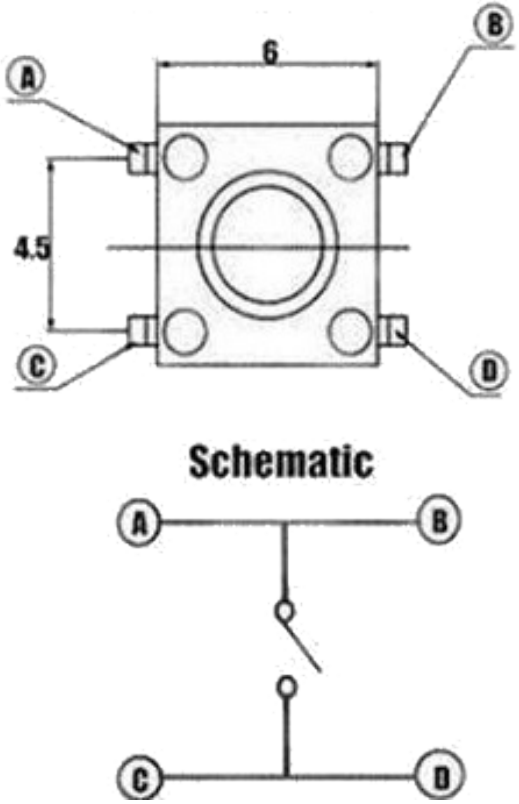


Resistencias



LED y botón



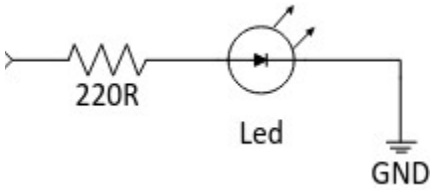


Recomendaciones

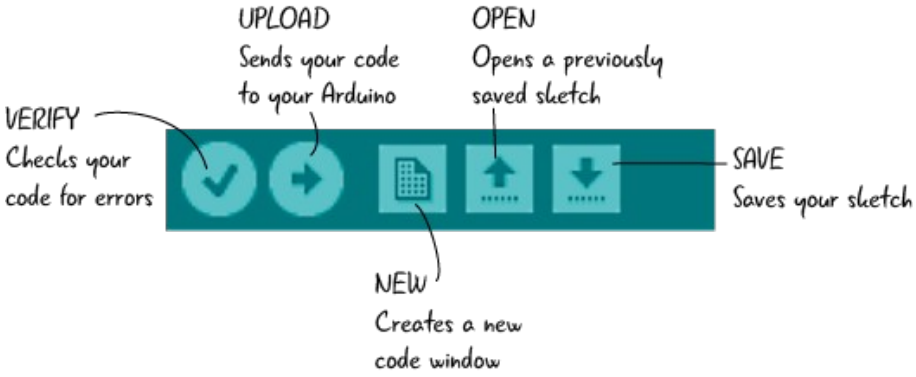
- Software
  - Guardar el código frecuentemente
  - Comentar el código
  - Aplicar buenas prácticas
- Hardware
  - Armar los circuitos sin alimentación
  - Comprobar conexiones antes de alimentar
  - No forzar conexiones
  - Comprobar componentes de forma aislada

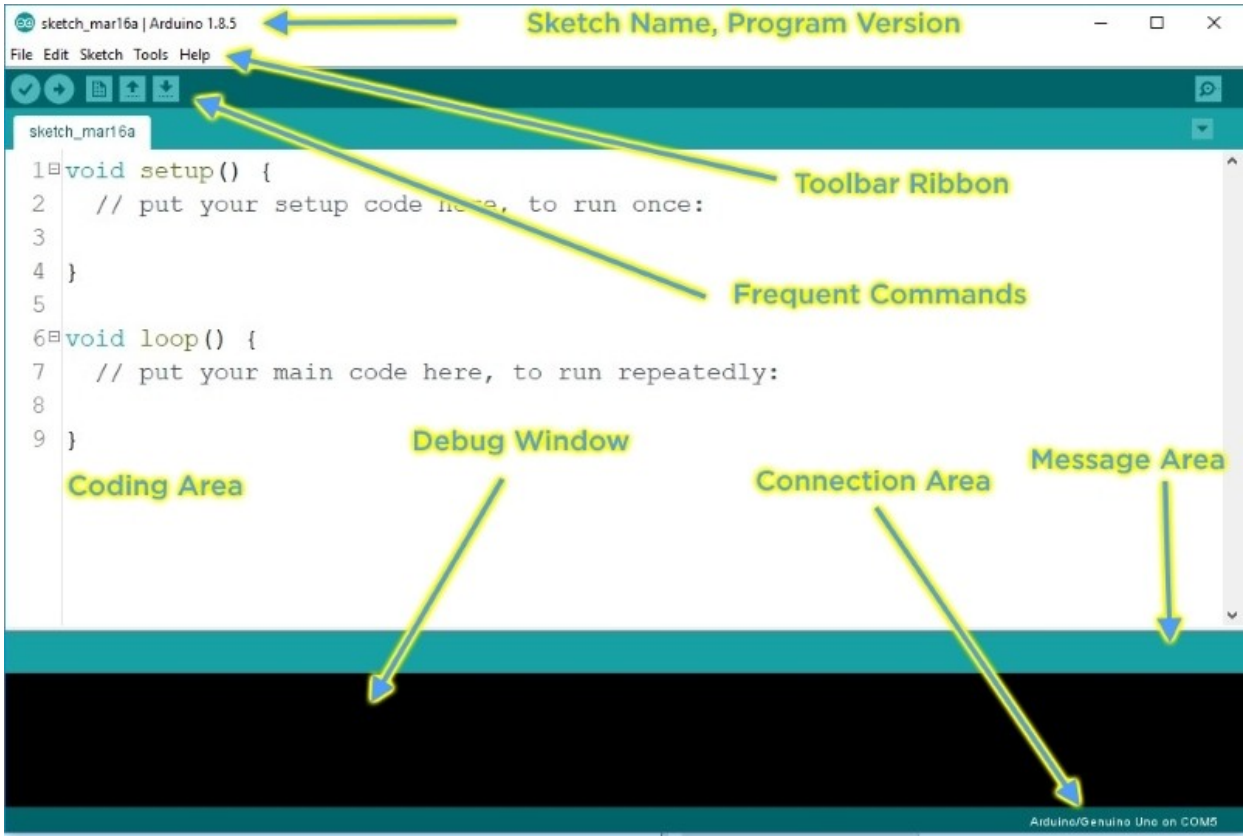
Tarea 1:: LED y Botón

- Cuidados:
  - NO cortocircuitar Vin y GND
  - Polaridad del LED y valor de resistencia (~160 Ohm)
- Desafíos:
  - prender el LED siempre
  - prender el LED cuando cuando se presione el botón.
- Materiales:
  - LED
  - Resistencias
  - Botón
  - Protoboard
  - Cables para interconexión



IDE Arduino





Setup()

- Se llama al comenzar
- Se utiliza para inicializar variables, los modos de los pines, librerias, etc.
- Corre una única vez, luego de prender o resetear la placa

loop()

- Se ejecuta luego del setup
- Como su nombre lo indica queda en loop

delay(ms)

- Pausa el programa la cantidad de tiempo indicada en milisegundos (ms).

pinMode(pin, mode)

- Configura el pin para funcionar como entrada o salida
- Parámetros:
  - pin: El número de pin arduino a modificar el modo
  - mode: Indica si el pin se va a setear como entrada (INPUT) o salida (OUTPUT).

digitalRead(pin)

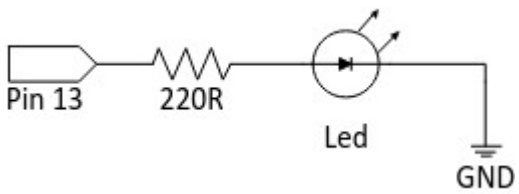
- Lee el valor del pin digital (HIGH o LOW).
- Parámetros:
  - Pin: El número de pin que se quiere leer

digitalWrite(pin,value)

- Escribe HIGH o LOW en un pin digital.
- Parámetros:
  - Pin: El número de pin que se quiere escribir
  - value: El valor que se quiere escribir

Tarea 2 :: Pender y apagar LED

- Previos:
  - Abrir el IDE Arduino
  - Configurar la placa a Arduino Uno (rev 3)
  - Configurar el puerto.
- Titilar LED en el pin 13 cada 1 segundo.
- Ademas titilar el LED en otro pin cada 2s.



- Pista: File → Examples → Basic → Blink

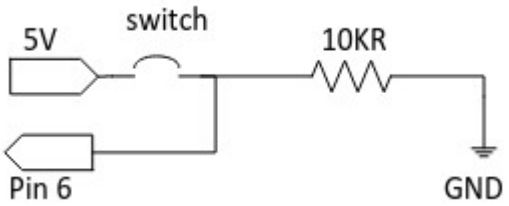
Sketch Tarea 2

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Tarea 3 :: Leer botón

- Leer botón con realimentación por serial

- Pista: Flie→Examples→Basics→DigitalReadSerial



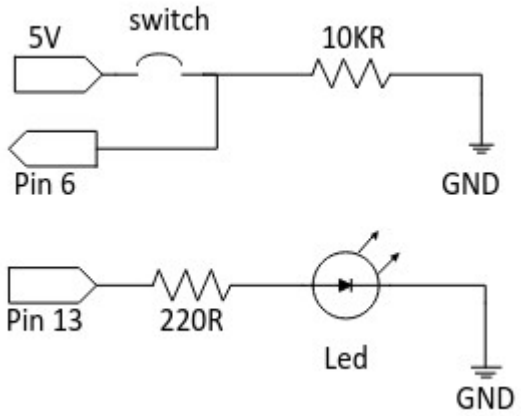
Sketch Tarea 3

```
// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);  // delay in between reads for stability
}
```

Tarea 4 :: LED y botón polling

- Prender si botón apretado
- Mantener el LED encendido por 2 s cada vez que se presiona el botón

- Pista: File→ Examples→ Digital→ Button

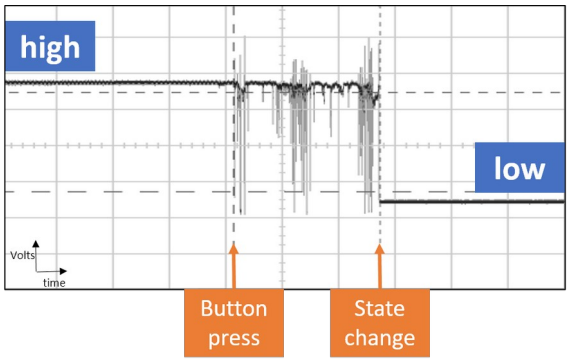


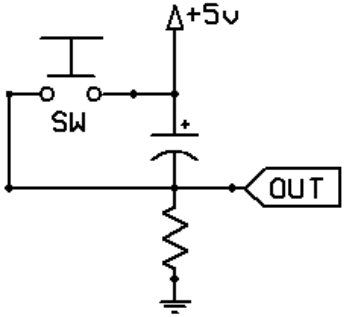
Sketch Tarea 3

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Tarea 5 :: Idem interrupciones

- Prender luz si botón se presiona usando interrupciones





- Variante: al presionar el botón mantener el LED encendido por N segundos.

Sketch Tarea 5

```
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
  attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), blinkLed, RISING);
}

void loop() {
  digitalWrite(LED_PIN, LOW);   // turn the LED off by making the voltage LOW
  delay(5000);
}

void blinkLed() {
  digitalWrite(LED_PIN, HIGH);
}
```

Biblioteca Timer1

- Abrir el gestor de bibliotecas del IDE de Arduino
- Instalar la Biblioteca desarrollada por Khoi Hoang
- Puede accederse al repositorio https://github.com/khoih-prog/TimerInterrupt_Generic

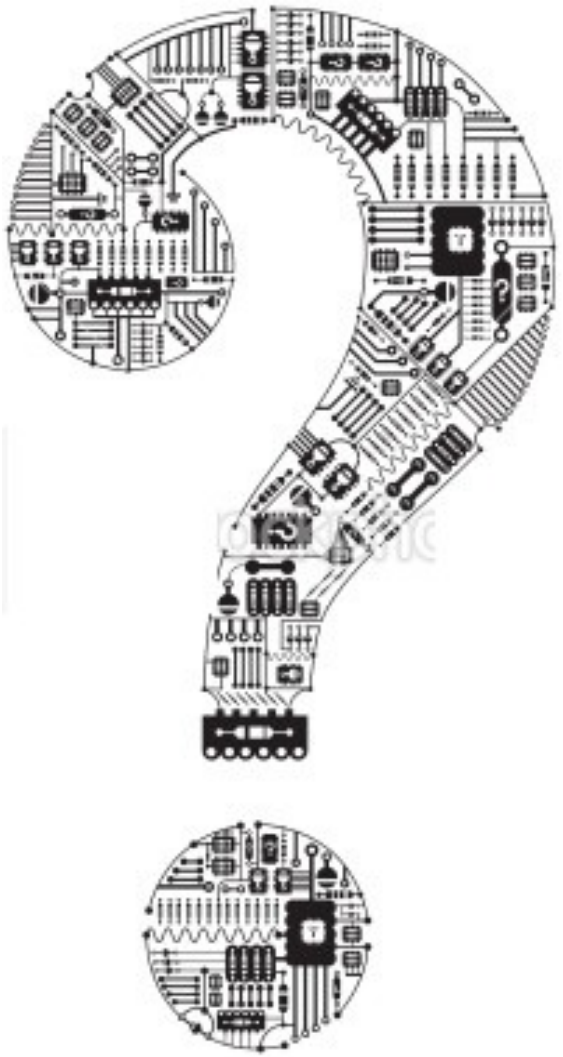
Sketch Tarea 5

```
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
  Timer1.initialize(100000); //Initialize timer with 100 ms
  Timer1.attachInterrupt(TimerHandler);
  attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), blinkLed, RISING);
}

void loop() {
  // nothing here!
}

void blinkLed() {
  digitalWrite(LED_PIN, HIGH);
  tics=0;
}

void TimerHandler(void) {
  tics++;
  if (tics>=20)
    digitalWrite(LED_PIN, LOW);
}
```

Tarea 6 :: Trabajo final

- Opción 1: Semáforo con botón para cruce peatonal
- Opción 2: Debouncing por software

Preguntas



mina
network management | artificial intelligence