



# Taller C++ orientado a Arduino

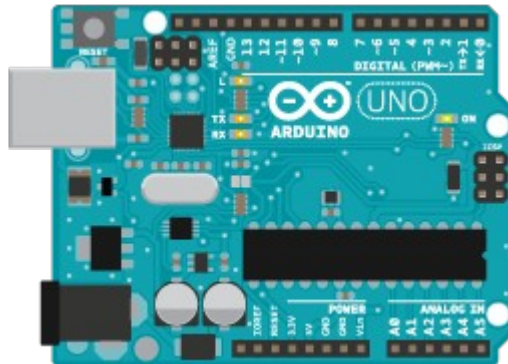
Sergio Esteban Pellejero  
Pablo Renero Balgañón

GUI – Grupo Universitario de Informática



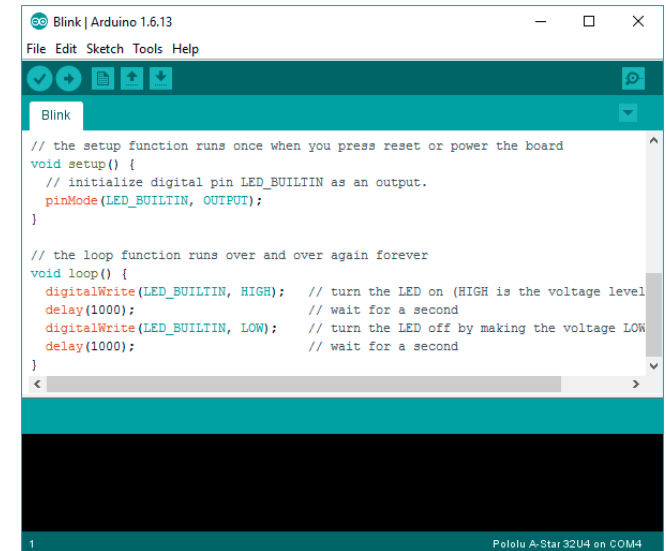
# 1. Historia

- Creado por estudiantes Italianos en 2005 ya que las placas de microcontroladores eran caras.
- Fines educativos.
- Actualmente se han vendido más de 250 mil placas en todo el mundo.
- Nombre en honor al Bar di Re Arduino, donde los estudiantes pasaban las horas muertas.



## 2. Lenguaje de Programación

- Arduino está basado en C++, por lo que nuestros propios programas sirven para arduino, con las modificaciones pertinentes.
- Nosotros por comodidad vamos a usar el IDE proporcionado por Arduino.
- El código de Arduino tiene limitaciones, no podemos escribir C++ puro, para Arduino.
- Al basarse en C++, el código tiene que ser compilado y enviado a nuestra placa de Arduino.



The screenshot shows the Arduino IDE interface with the 'Blink' sketch loaded. The code is as follows:

```
// Blink
// 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
}
```

The status bar at the bottom indicates 'Pololu A-Star32U4 on COM4'.



# 3. Consideraciones Previas

- Curso básico e introductorio.
- Materiales básicos:
  - > Placa de Arduino.
  - > Conector USB – serial Arduino.
  - > IDE o editor preferido con sus correspondientes extensiones.
  - > Hardware de electrónica general.

# 4. Primer Programa en Arduino

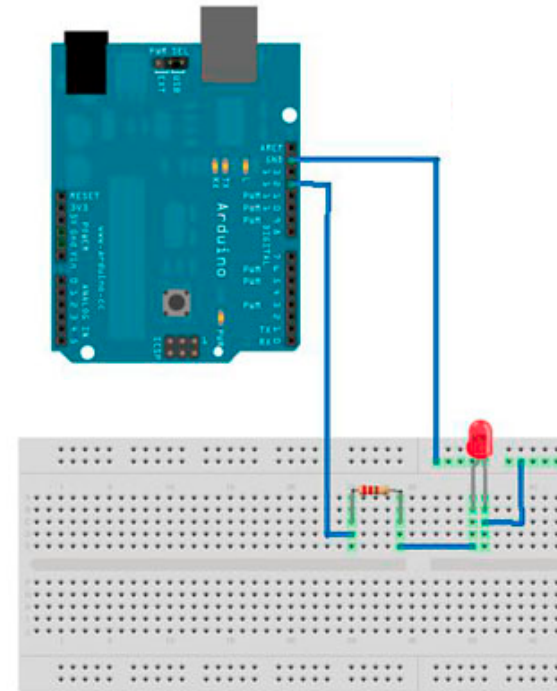
```
int led=13;

void setup () {
    pinMode(led, OUTPUT);
    Serial.begin(9600); //bps
    Serial.println("¡Hola Mundo!");
}

void loop {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000)
}
```

```
include <iostream>
using namespace std;

int main (void) {
    cout<<"¡Hola Mundo!"<< endl;
    return 0;
}
```



# 5. Tipos de datos

- Al igual que C++, Arduino tiene los siguientes tipos de datos básicos:
  - > Enteros: byte, int, word (exc), long.
  - > Decimales: float, double.
  - > Lógicos: boolean.
  - > Textos: char, string.
  - > Otros: void.
- Tipos de datos complejos (los mismos de C y C++):
  - > Arrays.
  - > Struct.
  - > Enum.
  - > Union.
  - > Typedef.
  - > Punteros.

## 6. Operadores

- Asignación: =
- Aritméticos: +, -, \*, /, %
- Lógicos: &, |, <, <=, ==, >=, >
- Bit: &&, >>, <<, ||
- Incrementos y decrementos: ++, --, +=, -=
- Casting: (tipoDatos) variable → (int) a
- Punteros: \*, &

# 7.1 Estructuras de control de flujo

## If ... else

```
If (condicion) {  
    // sentencias  
} else {  
    // sentencias  
}
```

## Switch case

```
switch (var) {  
    case label1:  
        // statements  
        break;  
    case label2:  
        // statements  
        break;  
    default:  
        // statements  
}
```



# 7.2 Ejemplo

## If ... else

```
int led=13;
Int cantidadLuz = 300;

void setup () {
    pinMode(led, OUTPUT);
    Serial.begin(9600); //bps
    Serial.println("¡Encendiendo bombillas!");
}

void loop {
    If (cantidadLuz >= 400) {
        digitalWrite(led, HIGH);
    } else {
        digitalWrite(led, LOW);
    }
    delay(1000);
}
```

## Switch case

```
int led=13;
Int cantidadLuz = 300;

void setup () {
    pinMode(led, OUTPUT);
    Serial.begin(9600); //bps
    Serial.println("¡Encendiendo bombillas!");
}

void loop {
    switch (cantidadLuz) {
        case 200:
            digitalWrite(led, LOW);
            break;
        case 300:
            digitalWrite(led, HIGH);
            break;
    }
    delay(1000);
}
```

# 8.1 Estructuras iterativas

## **for**

```
for (var; condicion;op) {  
    // sentencias  
}
```

## **while**

```
while (condicion) {  
    //sentencias  
}
```

## **do-while**

```
do {  
    // sentencias  
} while (condicion)
```

goto, continue, break

## 8.2 Ejemplo for

```
int timer = 100;    // The higher the number, the slower the timing.
```

```
void setup() {
```

```
    // use a for loop to initialize each pin as an output:
```

```
    for (int thisPin = 2; thisPin < 8; thisPin++) {
```

```
        pinMode(thisPin, OUTPUT);
```

```
    }
```

```
}
```

```
void loop() {
```

```
    // loop from the lowest pin to the highest:
```

```
    for (int thisPin = 2; thisPin < 8; thisPin++) {
```

```
        // turn the pin on:
```

```
        digitalWrite(thisPin, HIGH);
```

```
        delay(timer);
```

```
        // turn the pin off:
```

```
        digitalWrite(thisPin, LOW);
```

```
    }
```

```
// loop from the highest pin to the lowest:
```

```
for (int thisPin = 7; thisPin >= 2; thisPin--) {
```

```
    // turn the pin on:
```

```
    digitalWrite(thisPin, HIGH);
```

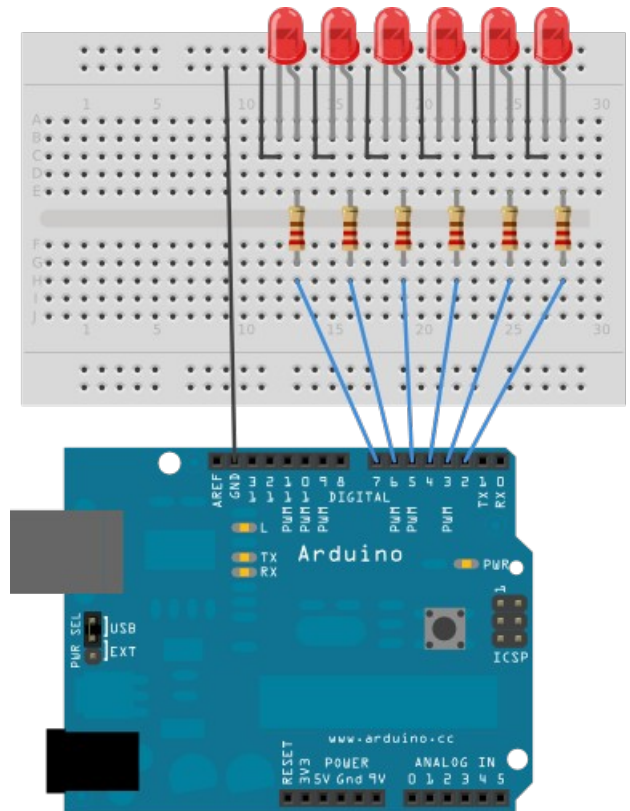
```
    delay(timer);
```

```
    // turn the pin off:
```

```
    digitalWrite(thisPin, LOW);
```

```
}
```

```
}
```



## 8.3 Ejemplos while y do-while

### while

```
int led=13;
Int cantidadLuz = 300;

void setup () {
    pinMode(led, OUTPUT);
    Serial.begin(9600); //bps
    Serial.println("¡Encendiendo bombillas!");
}

void loop {
    while (cantidadLuz >= 100) {
        digitalWrite(led, HIGH);
        cantidadLuz-=10;
    }
    delay(1000);
}
```

### Do - while

```
int led=13;
Int cantidadLuz = 300;

void setup () {
    pinMode(led, OUTPUT);
    Serial.begin(9600); //bps
    Serial.println("¡Encendiendo bombillas!");
}

void loop {
    do {
        digitalWrite(led, HIGH);
        cantidadLuz-=10;
    } while (cantidadLuz >= 100)
    delay(1000);
}
```

# 9. Funciones

```
void setup(){
  Serial.begin(9600);
}

void loop() {
  int i = 2;
  int j = 3;
  int k;

  k = myMultiplyFunction(i, j); // k now contains 6
  Serial.println(k);
  delay(500);
}

int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}
```

Misma declaración, uso y llamada que en C.

## Anatomy of a C function

Datatype of data returned,  
any C datatype.

Parameters passed to  
function, any C datatype.

"void" if nothing is returned.

Function name

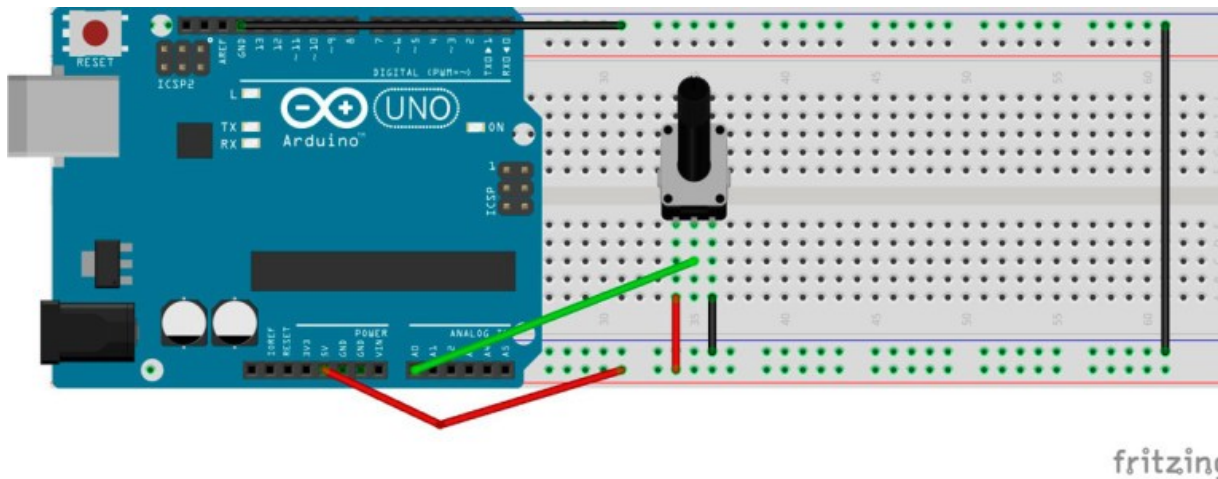
```
int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}
```

Return statement,  
datatype matches  
declaration.

Curly braces required.

# 10. Entrada Datos Arduino

```
int analogPin = 3;    // potentiometer wiper (middle terminal) connected to analog pin 3
                      // outside leads to ground and +5V
int val = 0;          // variable to store the value read
void setup()
{
  Serial.begin(9600);  // setup serial
}
void loop()
{
  val = analogRead(analogPin); // read the input pin
  Serial.println(val);         // debug value
}
```



**analogRead(int pin)** es una función que devuelve un número entero entre 0 y 1023, esto se debe a que mapea la lectura de valores de entre 0V y 5V de los sensores conectados a la entrada especificada