# "INTRODUCCIÓN A LA PROGRAMACIÓN EN ARDUINO Y TINKERCAD"

Trabajo Preparatorio N°2
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 este documento se revisará información relativa de la plataforma Arduino, sus diferentes modelos, los componentes que constituyen a dicha placa y su funcionamiento. De igualmanera, se trabajará con la plataforma Tinkercad y la introducción al proceso de programación en dicha plataforma.**

***Index Terms*—Arduino, memoria flash, pines, EEPROM.**

## I. Introducción

Arduino es una plataforma que hace uso de hardware libre, la circuitería usada hace que el precio sea bajo y de esa manera cualquier estudiante pueda adquirirla. Las características de hardware libre, programación de fácil entendimiento y gran capacidad hacen de Arduino una placa versátil y reusable ya que al terminar de usarla en cualquier aplicación es muy fácil retirar los respectivos periféricos conectados a esta placa y empezar a usar la placa para otra aplicación [1].

Además, los pines que esta placa posee hacen que la probabilidad de cometer un error de conexión sea muy baja.

## II. Objetivos

- Relacionar al estudiante con los fundamentos de los Sistemas Embebidos.
- Establecer e identificar características generales de la programación en Arduino.
- Explicar el funcionamiento de Tinkercad y Arduino IDE para diseñar e implementar códigos de programación. [2].

## III. Cuestionario

### *A. Consultar acerca de la placa de desarrollo Arduino Uno, sus componentes y su funcionamiento respectivo.*

Arduino Uno es una plataforma electrónica de código abierto (open-source) basada en una sencilla placa con entradas y salidas, en un entorno de desarrollo que está basado en el lenguaje de programación Processing. Es un dispositivo que conecta el mundo analógico con el digital.

Se puede encontrar un microcontrolador reprogramable, para que esta placa recoja las señales que se desean interpretar, se han colocado una serie de pines hembra. Asimismo, en el caso de que se desee obtener señales de esta placa se puede hacer uso de estos pines u otros pines macho que se encuentran incorporados en la placa. Para el hardware de la placa se hace uso de una placa electrónica PCB ya que esta es la forma m´as compacta y estable de construir un circuito. Esta placa contiene la circuitería suficiente para la elaboración de una serie de distintas aplicaciones. Se trata de que esta placa pueda ser alimentada con poca energía y que sea versátil. Hay que destacar que esta placa también es capaz de alimentar con 5 voltios y una baja corriente a cualquier dispositivo externo (por ejemplo, un ventilador pequeño capaz de enfriar la placa en el caso de ser necesario) [1].

El lenguaje de programación para esta plataforma, está basado en C++ ya que este es el de más sencilla comprensión para cualquier programador y a la vez tiene una gran capacidad de aplicación; otro beneficio adicional es la gran comunidad que existe alrededor de esta placa lo que permite poder solventar problemas que se encuentren en la realización de cualquier proyecto, también esta comunidad puede brindar ayuda e ideas en cuanto a la realización de proyectos con esta placa lo que le permite al usuario ser m´as eficiente en el desarrollo.

Se puede afirmar que esta plataforma fue creada en el año 2005 en el el Instituto de Dise˜no Interativo de Ivrea (Italia) donde se vio la necesidad de contar con un dispositivo que se pueda llevar a las aulas de clase y que este al alcance de cualquier estudiante. En un principio se tenía pensado usar esta placa exclusivamente en esta escuela, pero cuando el instituto tuvo que cerrar, se vio la necesidad de liberarlo y de que esa manera no se pierda el progreso que se obtuvo hasta el momento. Adem'as, al abrirlo al publico se pudo agregar mejoras al proyecto que se tenía pensado en un principio. Finalmente este pequeño proyecto es conocido alrededor del mundo y permite la enseñanza y la realización de proyectos dentro de institutos de educación superior. Actualmente existe una fundación internacional encargada del desarrollo de este proyecto [3].

Sus principales partes son:

- **Entradas:** Son los pines de nuestra placa que se pueden utilizar para hacer lecturas. En la placa Uno son los pines

digitales (del 0 al 13) y los analógicos (del A0 al A5).

- **Salidas:** Los pines de salidas se utilizan para el envío de señales. En este caso los pines de salida son sólo los digitales (0 a 13) [4].
- **Otros pines:** Se tiene pines como lo GND (tierra), 5V que proporciona 5 Voltios, 3.3V que proporciona 3.3 Voltios, los pines REF de referencia de voltaje, TX (transmisión) y RX (lectura) también usados para comunicación serial, RESET para resetear, Vin para alimentar la placa y los pines ICSP para comunicación SPI [3].
- **Alimentación:** Vin sirve para alimentar la placa pero lo más normal es alimentarlo por el jack de alimentación usando una tensión de 7 a 12 Voltios. También puede ser alimentado por el puerto USB pero en la mayoría de aplicaciones no se lo tiene conectado a un ordenador.
- **Comunicación:** Mediante USB para cargar los programas o enviar/recibir datos [4]. Sin embargo no es la única forma que tiene Arduino de comunicarse. Cuando se inserta una shield ésta se comunica con la placa utilizando los pines ICSP (comunicación ISP), los pines 10 a 13 (también usados para comunicación ISP), los pines TX/RX o cualquiera de los digitales ya que son capaces de configurarse como pines de entrada o salida y recibir o enviar pulsos digitales [5].
- **Shields:** Se llama así a las placas que se insertan sobre Arduino a modo de escudo ampliando sus posibilidades de uso. En el mercado existen infinidad de shields para cada tipo de Arduino. Algunas de las más comunes son las de Ethernet, Wi-Fi, Ultrasonidos, Pantallas LCD, relés, matrices LED's, GPS [4].

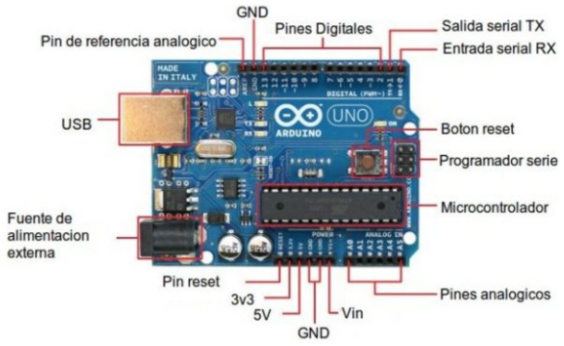



Fig. 1. Partes principales de Arduino

*B. Realizar una tabla comparativa (aplicaciones, consumo energético, dimensiones, interfaces, costo, entre otros) de los diferentes modelos de Arduino existentes actualmente en el mercado.*

La tabla solicitada se encuentra en el ANEXO 1.

*C. Consultar acerca de los diferentes tipos de datos (int, float, doble, char, entre otros) en Arduino*

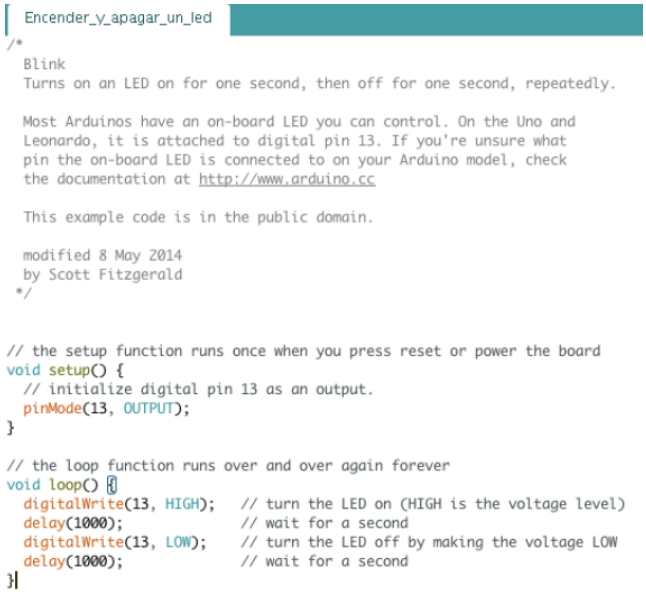
- **Tipo int:**Su tamaño es de 16 bits y representa un número con signo, entre -32768 y 32767. Este tipo es el más usado para variables de propósito general en Arduino, en los códigos de ejemplo que vienen con el IDE.
- **Tipo float:** Su tamaño es de 32 bits y representa un número con signo, entre 3.4028235E38 y 3.4028235E38. El Punto Flotante no es un tipo nativo en Arduino; el compilador debe realizar varios saltos para poder hacerlo funcionar [15].
- **Tipo double:**significa que la precisión de estos números es dos veces más que la precisión de los números del tipo float. En mayoría de los casos el tipo double es más cómodo. En muchos casos la precisión limitada de los números float simplemente es insuficiente. La razón de utilizar todavía el tipo float se encuentra en el ahorro de la memoria durante el almacenamiento (es importante para las grandes cantidades de matrices de números reales) [14].
- **Tipo char:** Su tamaño es de 8 bits y es un número con signo, entre -128 y 127. En algunos casos el compilador intentará interpretar este tipo de dato como un caracter, lo que puede generar resultados inesperados.
- **Tipo boolean:** Su tamaño es de 8 bits y puede tomar un valor lógico (verdad o falso).
- **Tipo byte:** Su tamaño es de 8 bits y es un número sin signo entre 0 y 255.
- **Tipo long:** Ocupa 32 bits (4 bytes), desde -2,147,483,648 a 2,147,483,647.
- **Tipo word:** Su tamaño es 16 bits y es un número sin signo entre, 0 y 65535 [16].
- **Tipo arreglos:** Son vectores que en cada uno de sus espacios contienen un tipo de dato. Hay que destacar que cada uno de estos arreglos empieza desde la posición cero.
- **Tipo punteros:** Son datos que contienen la dirección de un espacio de memoria. Son usados como accesos directos a espacios de memoria.

*D. Indique y defina brevemente cual es la estructura básica de un programa en Arduino.*

Una estructura b´asica de Arduino contiene los siguientes elementos:

- **Declaración de variables:** En este elemento se colocan todas las variables que se van a utilizar en la ejecución del programa. Las variables se declaran colocando el tipo de dato que esta va a hacer seguido del nombre que esta puede tener, esta variable puede tener un valor asignado desde este mismo punto o no. Antes de esta elemento se deben colocar las librerías que se van a usar.
- **setup():** La función setup() es llamada cuando el sketch empieza. Se usa para iniciar variables, modos de pin, empezar a usar librerías, etc. La funci´on setup() solo se ejecutará una vez, luego de cada vez que se encienda o reinicie la placa Arduino.
- **loop():**Después de crear la función setup(), que inicia y asigna los valores iniciales, la funci´on loop() hace precisamente lo que su nombre sugiere, es una funci´on que repite lo que esta dentro de ella mientras la placa

Arduino está encendida, permite al programa cambiar y responder. Se usa activamente para controlar la placa Arduino.



```
Encender_y_apagar_un_led
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Fig. 2. Código implementado en Arduino

## REFERENCES


[1] "Arduino - Tutorial 0: Introducción", Instructables. https://www.instructables.com/Arduino-Tutorial-0-Introducción/ (accedido nov. 22, 2020).

[2] E. Tatayo, "INTRODUCCIÓN A LA PROGRAMACIÓN EN ARDUINO Y TINKERCAD". C.P. SISTEMAS EMBEBIDOS, Accedido: nov. 22, 2020. [En línea].

[3] I. Mecafenix, "Arduino ¿Que es, como funciona? y sus partes", Ingeniería Mecafenix, abr. 25, 2017. https://www.ingmecafenix.com/electronica/arduino/ (accedido nov. 22, 2020).

[4] "Componentes para realizar proyectos con Arduino - Recursos - Picuino". https://www.picuino.com/es/arduprog/component-kit.html (accedido nov. 22, 2020).

[5] "Arduino: Definición, Componentes y Ejemplo Práctico", Aprende Ciencia y Tecnología, oct. 04, 2018. https://aprendecienciaytecnologia.com/2018/10/04/arduino-definicion-componentes-y-ejemplo-practico/ (accedido nov. 22, 2020).

[6] "Comparativa de todas las placas Arduino", ComoHacer.eu ¿Inventamos juntos? https://www.comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/ (accedido nov. 24, 2020).

[7] "Arduino - ArduinoBoardFio". https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardFio (accedido nov. 24, 2020).

[8] J. Ahedo, "Arduino 101 & Genuino 101 – Caracteristicas técnicas - Web-Robótica.com". https://www.web-robotica.com/arduino/placas-arduino/arduino-101-genuino-101-caracteristicas-tecnicas (accedido nov. 24, 2020).

[9] "Arduino Due", Components101. https://components101.com/microcontrollers/arduino-due (accedido nov. 24, 2020).

[10] "Arduino Ethernet Rev3 without PoE". https://store.arduino.cc/usa/arduino-ethernet-rev3-without-poe (accedido nov. 24, 2020).

[11] "Arduino Zero — Arduino Official Store". https://store.arduino.cc/usa/arduino-zero (accedido nov. 24, 2020).

[12] J. G. Cobo, "Arduino Zero Pro, la nueva placa del Proyecto Arduino", Hardware libre, abr. 06, 2015. https://www.hwlibre.com/arduino-zero-pro-la-nueva-placa-del-proyecto-arduino/ (accedido nov. 24, 2020).

[13] "ARM Cortex M0+ Based Arduino Zero Pro Board Gets Listed on Arduino.org". https://www.cnx-software.com/2015/03/05/arduino-zero-pro-arduino-org/ (accedido nov. 24, 2020).

[14] "GamePad using Android mobile sensors and Arduino", Arduino Project Hub. https://create.arduino.cc/projecthub/ashraf-nabil/gamepad-using-android-mobile-sensors-and-arduino-b0bf5c (accedido nov. 24, 2020).

[15] "Variables en Arduino — DIWO". http://diwo.bq.com/variables-en-arduino/ (accedido nov. 24, 2020).

[16] "Documentación para MQL5: Bases del lenguaje / Tipos de datos / Tipos reales (double, float)". https://www.mql5.com/es/docs/basis/types/double (accedido nov. 24, 2020).

# ANEXO I

TABLA I
COMPARACIÓN ENTRE LOS DIFERENTES TIPOS DE ARDUINO

| **Tipo** | **Microcontrolador** | **V[V]** | **Memoria Flash [KB]** | **EEPROM [KB]** | **# pines digitales** | **# pines analógicos** | **Uso** |
|---|---|---|---|---|---|---|---|
| *Uno* | Atmel ATmega320 de 8bit a 16MHz | 5 | 32 | 1 | 14 | 6 | Desarrollo de proyectos [8] |
| *TRE* | Atmel ATmega32u4 de 16MHz | 7 | 32 | 1 | 14 | 6 | Impresoras 3D, automatización de puertas y luces [9] |
| *101* | Intel Curie [6] | 7-12 | 196 | 8 | 14 | 6 | Reconocimiento de gestos [10] |
| *Zero* | Atmel SAMD21 de 48MHz | 3-5 | 256 | 16 | 14 | 6 | Contro de dispositivos electrónicos |
| *Zero Pro* | Cortex Mo+ de 48MHz [7] | 5-7 | 256 | 16 | 14 | 6 | Robótica o IoT |
| *Yun* | Atmel ATmega 32u4 de 16MHz | 5 | 32 | 1 | 20 | 12 | Conexión avanzada a redes y apps. |
| *Leonardo* | Atmel ATmega 32u4 de 16MHz | 5 | 32 | 1 | 20 | 12 | Elimina la necesidad de un procesador secundario [8] |
| *Due* | Atmel SAM3X8E ARM CortexM3 de 84MHz | 3.3 | 512 | 16 | 54 | 12 | Aplicaciones avanzadas debido a su alta velocidad [12] |
| *Mega* | Atmel ATmega2560 de 16MHz | 5 | 256 | 4 | 54 | 16 | Desarrollo de objetos interactivos. [13] |
| *Ethernet* | Atmel ATmega328 de 16MHz | 5 | 32 | 1 | 14 | 6 | Ethernet con PoE |
| *Fio* | Atmel ATmega328P de 8MHz | 7-12 | 32 | 1 | 14 | 8 | Aplicaciones inalámbricas |
| *Nano* | Atmel ATmega168 de16MHz | 7-12 | 32 | 1 | 14 | 8 | Encriptación de claves |
| *LilyPad* | Atmel ATmega168V y ATmega328V de 8MHz | 3-5 | 16 | 512 bytes | 14 | 6 | Integrado en prendas y textiles [13] |
| *Pro* | Atmel ATmega168 o ATmega328 de 8 y 16MHz | 3-5 | 16-32 | 1 | 14 | 6 | comunicación serie UART TTL [14] |