INTRODUCCIÓN A LA PROGRAMACIÓN POR BLOQUES DE ARDUINO EN TINKERCAD

Hugo Andrade, Marshall Flores
Universidad de las Fuerzas Armadas "E.S.P.E" Ingeniería en electrónica y Telecomunicaciones
hdandrad@espe.edu.ec, jmflores15@espe.edu.ec

Resumen: La utilización de Arduino hoy en día es muy demandada en proyectos de electrónica, ya que al ser una plataforma libre y de código abierto, los usuarios pueden modificar los requerimientos a sus necesidades, debido a que incorpora un microcontrolador y un entorno de desarrollo que permiten que todo esto sea posible.

En nuestro caso, al ser un entorno virtual en el cual vamos a trabajar, utilizaremos el Arduino UNO que incorpora Tinkercad, realizando la programación de un circuito mediante un entorno de desarrollo gráfico (Programación por bloques). El propósito de este trabajo de investigación es analizar los aspectos de hardware que ofrece un Arduino, haciendo énfasis en la entrada y salida de información. Esto se determinará a partir de la fundamentación correspondiente a los componentes que conforma un Arduino, su funcionalidad y aplicándolo a un circuito, en nuestro caso, a la operatividad de un sistema de semáforos, donde las entradas, corresponden al control de la modalidad en la que funcionan los mismos (modo habitual o modo precaución) y las salidas a los indicadores, los cuales ilustran que todo funcione correctamente.

Abstracto: hadware.- es la parte física de un ordenador o sistema informático. Está formado por los componentes eléctricos, electrónicos, electromecánicos y mecánicos, tales como circuitos de cables y luz, placas, memorias, discos duros, dispositivos periféricos y cualquier otro material en estado físico que sea necesario para hacer que el equipo funcione.

Introducción

Nuestro objetivo con este proyecto es implementar y analizar en base a la información de datos de entrada y salida de un Arduino, un circuito en el cual se muestre el funcionamiento del mismo, a través de la programación en un entorno gráfico y uso del Arduino en tinkercad, con el fin de entender el comportamiento del sistema de hardware que ofrece esta plataforma.

Posteriormente vamos a ver los diagramas aplicados en nuestro trabajo

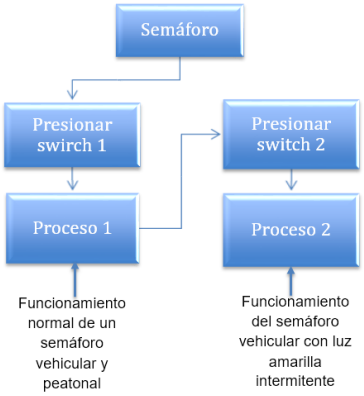


Diagrama 1.1 – Diagrama de bloques.

DIAGRAMA UML (CASO DE USO – CLASE)

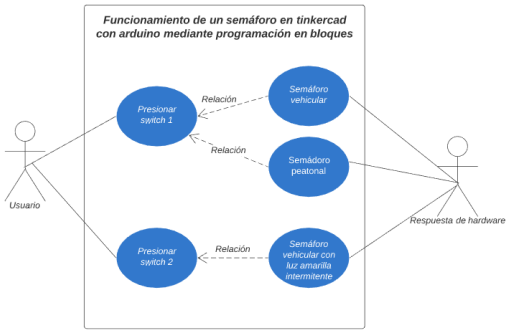


Diagrama 1.2 – Diagrama de uso.

DIAGRAMA ELÉCTRICO

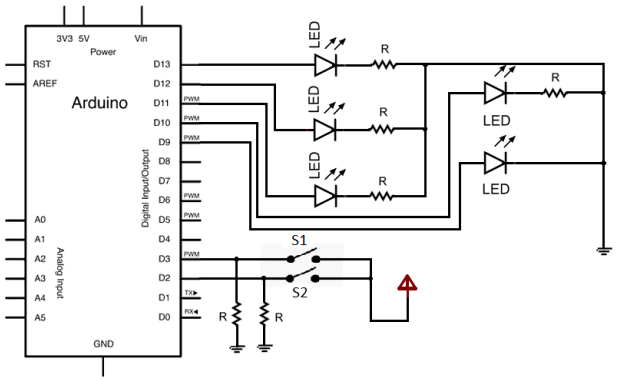


Diagrama 1.4 – Diagrama eléctrico con Arduino.

DIAGRAMA DE FLUJO

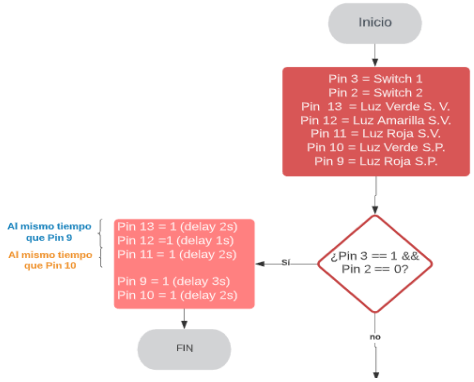


Diagrama 1.3 – Diagrama de flujo.

Los componentes y herramientas usadas fueron:

Componentes utilizados	
Software online TINKERCAD	Software en línea utilizado para realizar la simulación de un laboratorio virtual, cuya experiencia es muy similar a lo que se realizaría en un laboratorio físico. En este caso fue utilizado para representar el funcionamiento de un semáforo, mediante un Arduino, cuya programación se la elaboró mediante un entorno gráfico (programación por bloques).
Arduino	Es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. La placa en sí, está equipada con conjuntos de pines de E/S digitales y analógicas que pueden conectarse a varias placas de expansión y otros circuitos. En este caso, se lo programará mediante Tinkercad.
Protoboard	Placa de pruebas para realizar la conexión de todos los elementos que conforman un circuito
Resistencias	Componente electrónico utilizado para proteger a los demás elementos que conforman el circuito, oponiéndose al paso de la corriente
Dipswitch	Conjunto de interruptores eléctricos que se presenta en un formato encapsulado. Utilizado para modificar todas las combinaciones posibles de las entradas lógicas del circuito.
Diodos LED	Componente, cuya fuente de luz está constituida por un material semiconductor. En este caso es el equivalente a las luces de un semáforo. (Salidas)

Tabla 1.1 – Lista de componentes utilizados.

Luego procedimos a realizar un mapa de variables para ver de mejor forma la entrada y salida de datos para así ver el funcionamiento de nuestra programación en el Arduino

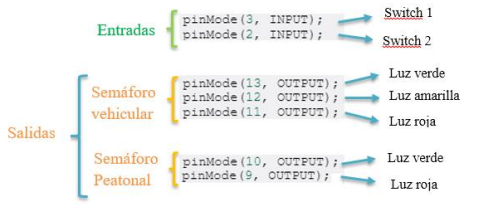


Figura 1.1 – Variables utilizadas.

La figura 1.1 nos ilustra de manera detallada todas las variables que han sido utilizadas para el correcto funcionamiento del programa e implementación del circuito (Semáforo) a través de un Arduino.
En la llave de color verde podemos observar las variables de entrada, las cuales corresponden a los switchs 1 (Pin Digital 3) y 2 (Pin Digital 2). Necesarios para realizar el control del funcionamiento habitual de un semáforo vehicular/peatonal y el funcionamiento "en modo precaución" de un semáforo vehicular (Luz amarilla intermitente) respectivamente.
La llave de color azul corresponde a las variables de salida que se han utilizado para representar las luminarias de cada semáforo, las tres primeras correspondientes al semáforo vehicular (Pin Digital 13 = Luz Verde | Pin Digital 12 = Luz Amarilla | Pin Digital 11 = Luz Roja) y las dos últimas al semáforo peatonal (Pin Digital 10 = Luz Verde | Pin Digital 9 = Luz Roja).

Interfaz para el usuario

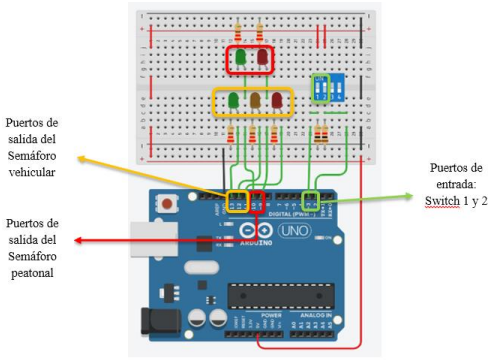


Figura 1.2 – Ejemplo - Variables visibles en la interfaz de usuario.

Donde:

- S1 → Switch 1 (Semáforos en "modo habitual"); → S2 → Switch 2 (Semáforos en "modo precaución").
- L.V.1 → Luz Verde (Semáforo vehicular); L.A.1 → Luz amarilla (Semáforo vehicular); L.R.1. → Luz Roja (Semáforo vehicular).

LV.2 → Luz Verde (Semáforo peatonal); L.R.2 → Luz Roja (Semáforo peatonal).

Semáforo		
Variables	Visibles	No visibles
S1 → Pin 1	X	
S2 → Pin 2	X	
L.V.1 → Pin 13	X	
L.A.1 → Pin 12	X	
L.R.1 → Pin 11	X	
L.V.2 → Pin 10	X	
L.R.2 → Pin 9	X	

Tabla 1.2 – Variables visibles y no visibles.

Ahora bien, necesitamos saber el tipo de cada variable que se ha utilizado, además de la función que realiza cada una en el programa. Esto se ilustra en la tabla 1.3.

Semáforo		
Variables	Tipo	Función
S1	Booleano	Variable de entrada encargada de controlar el funcionamiento habitual de un semáforo peatonal y vehicular.
S2	Booleano	Variable de entrada encargada de controlar el funcionamiento en "modo precaución" de un semáforo vehicular.
L.V.1	Booleano	Variable de salida que representa al encendido de la luz verde de un semáforo vehicular en un lapso de 2 segundos.
L.A.1	Booleano	Variable de salida que representa al encendido de la luz amarilla de un semáforo vehicular en un lapso de 1 segundo.
L.R.1	Booleano	Variable de salida que representa al encendido de la luz roja de un semáforo vehicular en un lapso de 2 segundos.
L.V.2	Booleano	Variable de salida que representa al encendido de la luz verde de un semáforo peatonal en un lapso de 3 segundos. (Mientras L.A.1 y L.R.1 estén en 1 lógico)
L.R.2	Booleano	Variable de salida que representa al encendido de la luz roja de un semáforo peatonal en un lapso de 2 segundos. (Mientras L.V.1 esté en 1 lógico)

Tabla 1.3 – Variables visibles y no visibles.

EXPLICACIÓN DEL CÓDIGO FUENTE

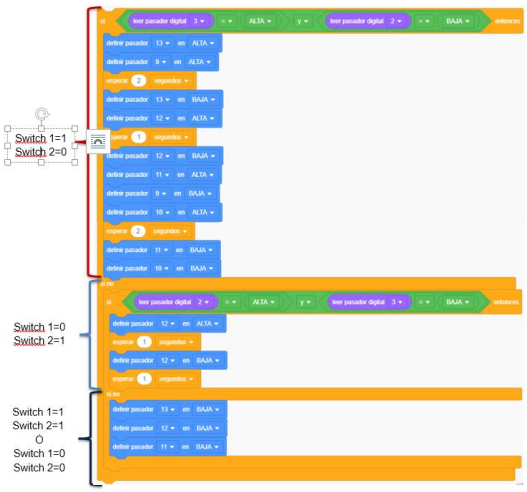


Figura 1.3 – Detección de errores mediante paridad par.

Programación de un Arduino en Tinkercad mediante bloques

El funcionamiento del programa básicamente consiste de estructuras condicionales "If-Else" anidadas, en las que se postulan las diferentes condiciones para que los semáforos funcionen en "modo habitual" o "modo precaución". (Figura 1.3)

Presionando S1 (Switch 1)

La primera estructura condicional nos estipula que al momento de presionar el switch 1, el programa ejecutará la secuencia de bloques (Ilustrada en la figura 1.3 – Llave de color rojo), en la que se resuelve todo el algoritmo correspondiente al funcionamiento de un semáforo vehicular y peatonal en su "modo habitual". Esto quiere decir que la luz verde del semáforo vehicular se encenderá durante dos segundos, debido a que el pasador (pin digital) 13 se encuentra en estado lógico "ALTO" con una espera de 2 segundos. Finalizado el transcurso de este tiempo se apagará y a su vez se encenderá la luz amarilla (pin digital 12) durante 1 segundo. Simultáneamente la luz roja del semáforo peatonal (pin digital 9) se enciende hasta que transcurran los tres segundos, en los cuales ocurrió la primera parte del proceso.

Finalmente, la luz amarilla del semáforo vehicular (pin digital 12) y la luz roja del semáforo peatonal (pin digital 9) se posicionan en el estado lógico "BAJO". Y la luz roja del semáforo vehicular (pin digital 11) junto con la luz verde del semáforo peatonal cambian su estado lógico a "ALTO" durante 2 segundos. Acabado este proceso, el programa se repetirá hasta cambiar el estado lógico del switch 1.

Presionando S2 (Switch 2)

La segunda estructura condicional nos estipula que al momento de presionar el switch 2, el programa ejecutará la secuencia de bloques (Ilustrada en la figura 1.3 – Llave de color celeste), en la que se resuelve el algoritmo correspondiente al funcionamiento de un semáforo vehicular en "modo precaución" que consiste de un total de 4 bloques de código; en el cual, el primer bloque coloca al pin digital 12 (luz amarilla del semáforo vehicular) en estado lógico "ALTO" durante 1 segundo (segundo bloque) y posterior a ello el mismo pin digital cambia al estado lógico "BAJO" (tercer bloque) durante 1 segundo (cuarto bloque). Acabado este proceso, esta secuencia se repetirá hasta cambiar el estado lógico del switch 2.

Switch 1 y 2 en "ALTO" o en "BAJO"

Si no se cumple ninguna de las combinaciones anteriores, se ejecutarán los tres últimos bloques de código (Figura 1.3 – Llave de color azul). Los cuales representan al apagado de todo el sistema de semáforos, ya que los pines digitales 11, 12 y 13 cambian a estado lógico "BAJO".

Una de las ventajas de programar un Arduino en Tinkercad, es que además de elaborar el programa mediante bloques podemos visualizar el mismo código de manera "textual". Y de esta manera tener dos opciones de realización para el programador, dependiendo de cual le resulte más intuitiva, A continuación, mostramos el código que se generó de manera automática en Tinkercad (Figura 1.4).

```
void setup()
{
  pinMode(3, INPUT);
  pinMode(2, INPUT);
  pinMode(13, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
}

void loop()
{
  if (digitalRead(3) == HIGH && digitalRead(2) == LOW) {
    digitalWrite(13, HIGH);
    digitalWrite(9, HIGH);
    delay(2000); // Wait for 2000 millisecond(s)
    digitalWrite(13, LOW);
    digitalWrite(12, HIGH);
    delay(1000); // Wait for 1000 millisecond(s)
    digitalWrite(12, LOW);
    digitalWrite(11, HIGH);
    digitalWrite(9, LOW);
    digitalWrite(10, HIGH);
    delay(2000); // Wait for 2000 millisecond(s)
    digitalWrite(11, LOW);
    digitalWrite(10, LOW);
  } else {
    if (digitalRead(2) == HIGH && digitalRead(3) == LOW) {
      digitalWrite(12, HIGH);
      delay(1000); // Wait for 1000 millisecond(s)
      digitalWrite(12, LOW);
      delay(1000); // Wait for 1000 millisecond(s)
    } else {
      digitalWrite(13, LOW);
      digitalWrite(12, LOW);
      digitalWrite(11, LOW);
    }
  }
}
```

Datos de entrada y salida

Desarrollo del programa

Figura 1.3 – Detección de errores mediante paridad par.

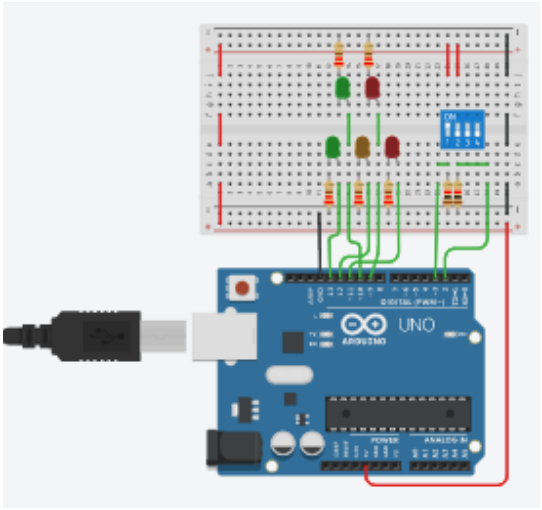
DESCRIPCIÓN DE PRERREQUISITOS Y CONFIGURACIÓN

En lo que corresponde a la utilización de aplicaciones secundarias para que el programa funcione correctamente, podemos decir que no se ha necesitado de ninguna que influya directamente sobre su programación, ya que Tinkercad ofrece formatos de programación tanto en un entorno textual como gráfico.

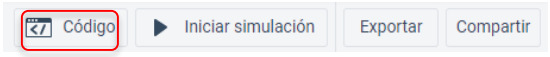
Ahora bien, necesitamos saber cuales son los pasos para poder programar un Arduino en una interfaz gráfica (mediante bloques) en Tinkercad. Para lo cual, debemos:

1. Seleccionar en la sección de componentes, todos los elementos que se utilizarán en el circuito, entre ellos, el Arduino UNO R3 que incorpora Tinkercad

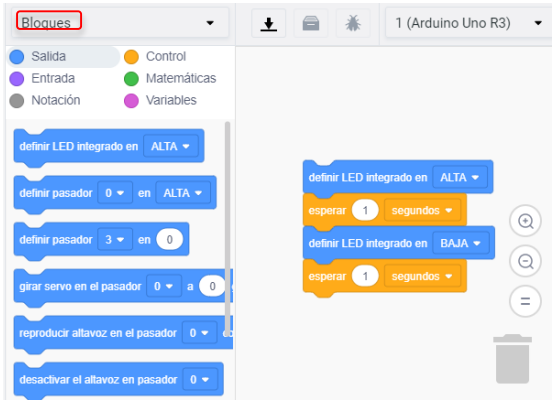
2. Conectar todos los componentes electrónicos de entrada y salida en los pines digitales del Arduino, se recomienda utilizar un protoboard para mayor facilidad al momento de realizar las conexiones.



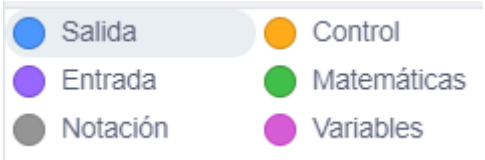
3. Dirigirse a la barra de control y seleccionar la opción “Código”



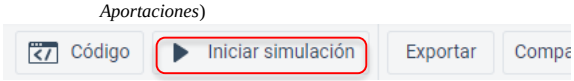
4. Nos encontraremos con la siguiente interfaz y seleccionaremos la opción de programación mediante “Bloques”. Por default el programa incorpora 4 bloques de código con instrucciones variadas. Entonces procedemos a borrar y a programar nuestro circuito



5. En la siguiente figura se encuentran todos los bloques categorizados (bloques de salida, entrada, notación, control, matemáticas, variables). Entonces seleccionamos, los que sean necesarios y de esta manera dar las instrucciones para que nuestro circuito funcione a nuestra necesidad. En este caso un sistema de semáforos controlados por dos switch de entrada



6. Una vez que ya se ha realizado la programación (ilustrada en la sección *8. Explicación del código fuente*) procedemos a dar clic en “Iniciar simulación” y verificar que el circuito funcione correctamente (Sección 9. *Aportaciones*)



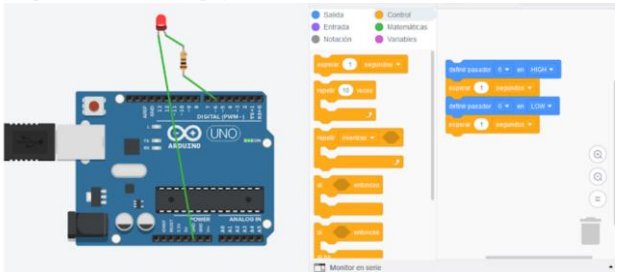
Por otro lado, para el correcto funcionamiento del circuito, hay que tener en cuenta varios puntos específicos. Los cuales son indispensables para no tener ningún inconveniente al momento de ejecutar el circuito.

- Los semáforos en “modo habitual” solamente funcionan cuando la entrada (switch 1) se encuentre en estado lógico “ALTO”.
- El semáforo vehicular en “modo precaución” solamente funciona cuando la entrada (Switch 2) se encuentre en estado lógico “ALTO”.
- Si los dos switch se encuentran en estado lógico “ALTO o “BAJO”

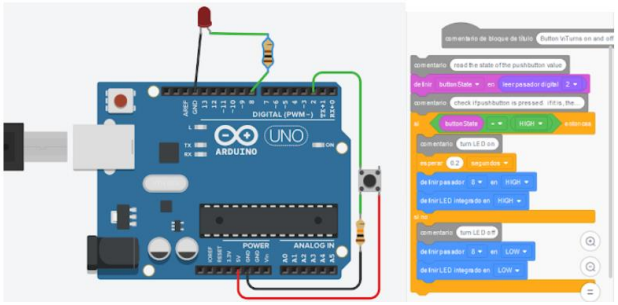
(S1=1 && S2=1) || (S1=0 && S2=0) las salidas permanecerán en estado lógico “BAJO”

Nota: Para programar en Arduino mediante Tinkercad, se requiere de la familiarización con fundamentos básicos de programación (Principalmente sobre estructuras condicionales, lógica booleana) y conocimientos sobre la constitución, funcionalidad y partes de un Arduino.

Ejemplo del encendido y apagado de un LED en Tinkercad



Ejemplo del encendido y apagado de un LED tres veces consecutivas, mediante un pulsador en Tinkercad



Ejemplo del uso y funcionamiento del circuito de semáforos implementado

- **Presionando Switch 1 (Semáforos en “Modo habitual”)**

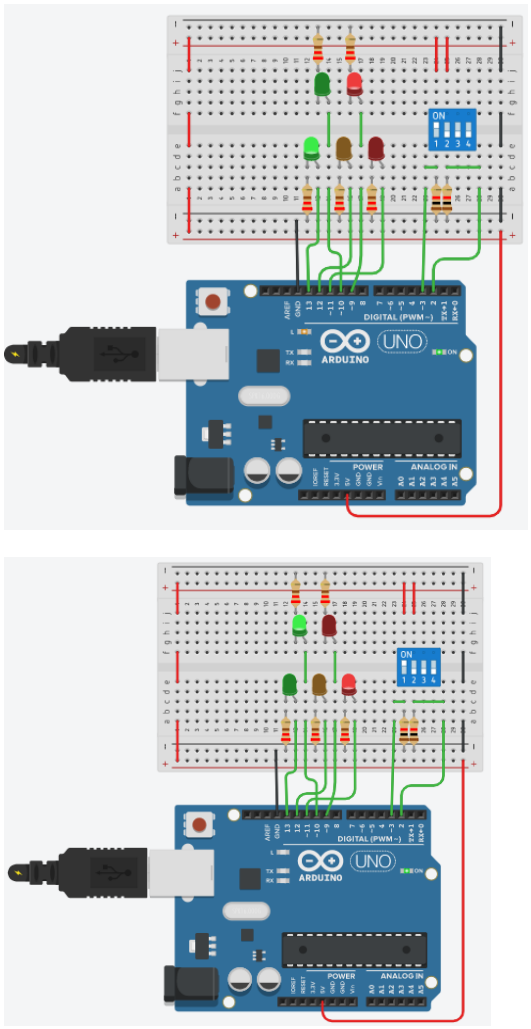


Figura 2.1 – Ejemplo Parida Par.

- **Presionando Switch 2 (Semáforo en "Modo Precaución")**

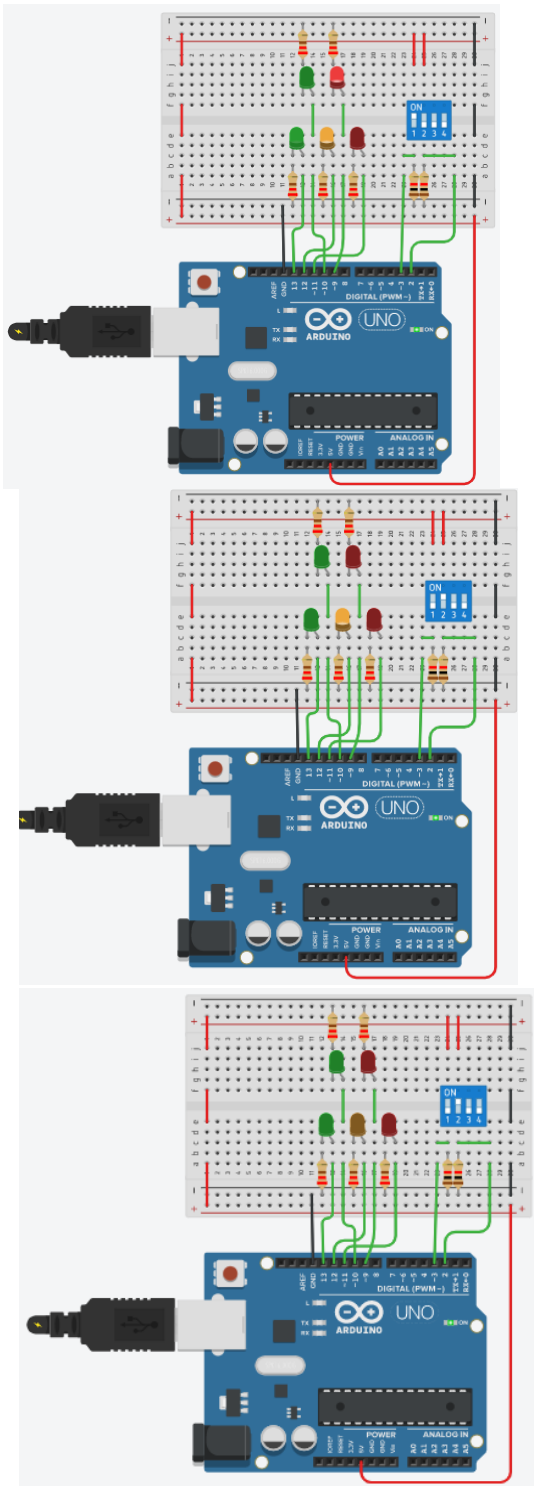
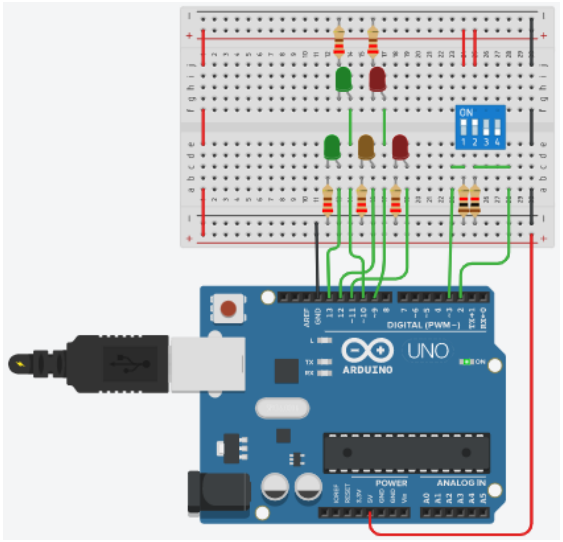
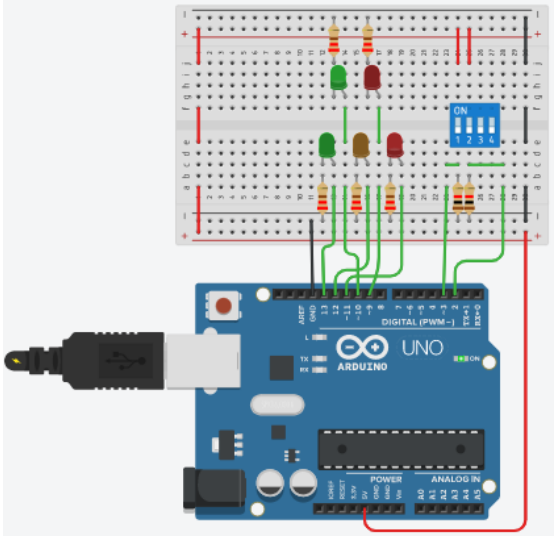


Figura 2.1 – Ejemplo Parida Impar.

- **Switch 1 y 2 en "ALTO" o "BAJO" (Semáforos en "Modo Inactivo")**





CONCLUSIONES

- El diseño e implementación del programa, se lo ha realizado de manera que cumpla el objetivo de realizar la detección de error de un número binario, mediante paridad par e impar. Todo este proceso se llevó a cabo con la programación en un entorno gráfico que nos ofrece MIT APP INVENTOR 2, el cual nos permitió realizar la aplicación para que cualquier usuario, pueda utilizarlo desde su dispositivo móvil. Tal como se ha podido observar en las diferentes figuras del presente informe, el diseño de la app es muy intuitiva y amigable al usuario.
- La detección de errores por paridad par o impar es muy esencial para sistemas de comunicación digital, los cuales necesitan de valores precisos y sin equivocaciones. La manera de entender cómo funciona cada paridad es muy sencilla. En el caso de paridad par, tendríamos un bit de paridad de 0 si la suma total de 1's es par, es decir, no existiría error. Caso contrario el bit de paridad tendría el valor de 1 y por lo tanto existiría error. Por otro lado, en el caso de paridad impar, tendríamos un bit de paridad de 0 si la suma total de 1's es impar. Caso contrario el bit de paridad tendría el valor de 1 si la suma total de 1's es par. Cada uno de los conceptos de los métodos de detección de error mencionados, fueron muy importantes y necesarios para la realización del algoritmo, el cual fue implementado en App Inventor.
- La realización del programa se la realizó en base a todo lo investigado. Prácticamente necesitamos realizar el conteo de 1's, y mediante el método de paridad que se escoja, se determinará si existe error dependiendo si la suma mencionada es par o impar. De esta manera logramos diseñar un algoritmo, el cual se basó principalmente en divisiones sucesivas para realizar el conteo de cifras, la suma total de 1's y por último encontrar el bit d paridad que determinó el error. Todo esto, mediante el uso de la estructura repetitiva While. Cabe mencionar que todo el algoritmo se lo ha hecho de manera correcta, incluso utilizamos un software de apoyo (Java NetBeans) para realizar las diferentes comprobaciones y pruebas que el programa requería. Y al momento de realizar la investigación con los artículos de los autores investigados, podemos darnos cuenta de la gran similitud y relación que encontramos con respecto a nuestro trabajo de investigación, todo esto se lo ha detallado en sección de Estado del arte.

BIBLIOGRAFÍA

[1] JC Martinez-Santos, O.Acevedo Patiño y SH Contreras-Ortiz, «Sci-Hub IEEEXplore,» 29 Noviembre 2017. Available: https://sci hub.tw/https://ieeexplore.ieee.org/document/8123929

[2] F. J. Jiménez, F. R. Lara y M. D. Redel, «Sci-Hub IEEExplore,» 08 Septiembre 2014. Available: https://sci-hub.tw/https://ieeexplore.ieee.org/document/6893988

[3] M.S. Perez y E. Carrera, «Sci-Hub IEEEXplore,» 21 Febrero 2015. Available: https://sci-hub.tw/https://ieeexplore.ieee.org/document/7055564

Anónimo. (12 de Febrero de 2016). descubrearduino.com. Recuperado el 25 de Julio de 2020, de descubrearduino.com: https://descubrearduino.com/arduino-uno/

F, Y. (3 de Agosto de 2018). Xataka. Obtenido de Xataka: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno