



CURSO:

Teoría de la Computación

BLOQUE:

FC-SMVISF-AD03A01M

DOCENTE:

RODRIGUEZ URQUIAGA, ROBERTO JOSUE

INTEGRANTES

-Valerio Crisostomo, Fabrizio Rafael (2211061) (100%)

-Jara Lamas, Claudio Cronwell (2110669) (100%)

- Jara Leon, Joaquin Esteban (2121231) (100%)

Lima-Perú

2024

Índice

1. Introducción.....	3
2. Marco Teórico.....	3
2.1 Autómatas.....	4
2.3 Tipos de Autómatas.....	4
2.3.1 Autómatas Finitos Deterministas (DFA).....	4
2.3.2 Arduino.....	4
2.3.3 Placas Arduino.....	4
3. Trabajos relacionados.....	5
4. Objetivo Principal:.....	8
5. Objetivos Secundarios:.....	8
6. Diseño de estudio.....	8
6.1 Descripción del Sistema.....	8
6.2 Interacción entre componentes:.....	9
7. Diagrama de estado y transiciones:.....	9
7.1 Diagrama de Transiciones:.....	9
7.2 Diagrama de estados:.....	11
8. Resultados:.....	12
9. Implementación:.....	13
10. Conclusiones:.....	17
11. Referencias:.....	18
12. Anexos.....	18

“Sistema de Control de Semáforos Basado en Autómatas”

1. Introducción

El crecimiento exponencial del tráfico en las ciudades modernas ha llevado a la necesidad de soluciones más eficientes y tecnológicamente avanzadas para la gestión del tráfico.

Actualmente, uno de los principales desafíos que enfrentan las áreas urbanas es la congestión vehicular, que causa demoras en los trayectos, aumento en el gasto de combustible, emisiones contaminantes y un mayor riesgo de accidentes. En este contexto, los avances en la teoría de la computación, particularmente en el uso de autómatas finitos y máquinas de Turing, ofrecen una oportunidad significativa para abordar estos problemas.

La teoría de la computación, desarrollada en gran parte gracias a los trabajos pioneros de Alan Turing y John Von Neumann, se enfoca en analizar qué problemas pueden resolverse computacionalmente y con qué eficiencia. Utilizando modelos como los autómatas finitos, las máquinas de Turing y las computadoras cuánticas, esta teoría estudia lógicas y lenguajes formales para razonar sobre programas, distinguiendo entre problemas computables y no computables, y clasificando los primeros según su complejidad.

Este proyecto tiene como objetivo principal investigar la teoría de la computación con un enfoque especial en el uso de autómatas para modelar y regular el funcionamiento de un sistema semafórico. La finalidad es mejorar la gestión del tráfico en una intersección compuesta por cuatro calles, abordando las insatisfacciones de transeúntes y conductores que sufren demoras, gastos excesivos de combustible, y enfrentan mayores riesgos de accidentes. La implementación de un sistema eficiente para el control de semáforos puede mejorar significativamente el flujo de vehículos y peatones, asegurar la seguridad vial y reducir la congestión de manera notable.

2. Marco Teórico

Para el actual proyecto tenemos que tener en cuenta ciertos temas de autómatas para poder entender de una mejor forma el actual proyecto.

Empezaremos comprendiendo que son los autómatas y los autómatas a utilizar en este proyecto y luego sobre el microcontrolador Arduino .

2.1 Autómatas

Los autómatas son modelos matemáticos de sistemas de procesamiento de información que pueden describirse de manera precisa y sistemática. Un autómata se compone de un conjunto finito de estados, un alfabeto de símbolos de entrada, una función de transición que determina el cambio de estados en respuesta a los símbolos de entrada, un estado inicial y un conjunto de estados finales o de aceptación.

2.3 Tipos de Autómatas

2.3.1 Autómatas Finitos Deterministas (DFA)

Los DFA son los autómatas más simples y consisten en un conjunto finito de estados donde cada estado tiene una transición definida para cada símbolo del alfabeto de entrada. Son utilizados principalmente para el reconocimiento de patrones y la implementación de analizadores léxicos en compiladores.

2.3.2 Arduino

Arduino es una plataforma de hardware libre «open-source» basada en una placa con un microcontrolador y un entorno de desarrollo integrado (IDE) que facilita la escritura de software para la placa. Fue diseñada para ser accesible y fácil de usar, incluso para personas sin experiencia previa en electrónica o programación.

2.3.3 Placas Arduino

Las placas Arduino vienen en diversas formas y tamaños, cada una adaptada a diferentes tipos de proyectos. Algunas de las placas más comunes incluyen:

Arduino Uno: La placa más popular, ideal para principiantes.

Arduino Mega: Ofrece más pines de entrada/salida y memoria que la Uno.

Arduino Nano: Más pequeña y adecuada para proyectos con limitaciones de espacio.

Arduino Leonardo: Puede actuar como un dispositivo USB nativo, como un teclado o ratón.

3. Trabajos relacionados

.- López Laguna, J. M. (2010). *Simulador de flujo de tráfico basado en autómatas celulares* (Master's thesis).

Descripción: Este artículo nos describe un proyecto que desarrolla un simulador de flujo de tráfico utilizando autómatas celulares en Java. La aplicación cuenta con una interfaz visual que permite a los usuarios interactuar con el sistema y visualizar el entramado del tráfico. Este entramado se representa mediante una matriz de celdas cuyo estado cambia según su estado actual y el de las celdas vecinas.

La interfaz permite a los usuarios diseñar sus propios mapas, añadir o eliminar elementos en cualquier momento, y realizar pruebas y simulaciones según su criterio. El sistema proporciona herramientas para realizar análisis exhaustivos de los resultados de las simulaciones, permitiendo comparar diferentes estructuras para determinar cuál es más favorable para un tráfico fluido.

Todos los datos relacionados con las simulaciones y los mapas creados se almacenan en una base de datos. Esto facilita la comparación y el análisis de simulaciones anteriores con las nuevas, ya que se pueden cargar y comparar fácilmente con las nuevas simulaciones realizadas.

2.- López, I. (Diciembre 2011). Control de un Cruce de Semáforos con Autómata Programable ://www.ugr.es/~iloes/proyectos/fisicayelectronica/SemPLC.pdf

Descripción: Este es un artículo sobre el control de un semáforo con un autómata programable. Se discuten el hardware y el software necesarios, así como los pasos de programación. El sistema utiliza un PLC Siemens SIMATIC S7-300. El PLC está programado con el software STEP 7. El programa consta de varios segmentos que se ejecutan en un ciclo. El tiempo de ciclo es de 18 segundos. El PLC controla los semáforos encendiéndose y apagándose por pares. Las luces están temporizadas para que no haya conflicto entre el tráfico que va en diferentes direcciones.

3.- Costa, R. J. M., & dos Anjos Rosa, J. L. (2015). Estudo de linguagens formais: conceitos e prática aplicados na construção de um semáforo. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 3(1).

Descripción: Este artículo destaca por la explicación y la creación de un autómata para la creación de un semáforo. En este se ha desarrollado un semáforo con tres estados: Abierto (verde), Atención (amarillo) y Cerrado (rojo), utilizado como sistema de control de tráfico. Cuando el semáforo muestra "Abierto", permite el paso de vehículos; "Atención" indica que los vehículos deben reducir la velocidad y detenerse eventualmente; y "Cerrado" indica que los peatones pueden cruzar. El cambio de estado del semáforo ocurre en intervalos de tiempo específicos: 10 segundos de "Abierto" a "Atención", tres segundos de "Atención" a "Cerrado" y nuevamente 10 segundos de "Cerrado" a "Abierto". Además de estos cambios programados, los peatones pueden solicitar la transición al estado "Cerrado". Si la solicitud ocurre cuando el semáforo ya está en "Cerrado", se añaden cuatro segundos al tiempo de transición. Si la solicitud ocurre en "Atención", no tiene efecto; y si ocurre en "Abierto", se reducen cuatro segundos del tiempo de transición (si quedan menos de cuatro segundos, el sistema cambia de estado inmediatamente).

El sistema solo permite una solicitud por estado y luego vuelve a los tiempos de transición programados. El semáforo opera mediante una máquina de estado con los estados mencionados y las transiciones entre ellos ocurren por tiempo predeterminado o por solicitud del peatón.

4.- Fuentes, V. M. (2014). Introducción a la plataforma Arduino y al Sensor ultrasónico HC-SR04. *Profesional, Universidad Carlos III de Madrid*.

Descripción: En este artículo nos habla sobre la utilización de los sensores HC-SR04. Este documento se divide en varias secciones en las cuales nos hablan sobre arduino y la utilización de de los sensores. En la sección titulada "Arduino", nos da una visión general de la plataforma Arduino, explicando sus características principales tanto a nivel de hardware como de software.

En el apartado de Sensor ultrasónico, se explicará el concepto de sensor y el concepto de ultrasonidos. Se detallarán las características y el funcionamiento interno del sensor de ultrasonidos utilizado en el proyecto, el sensor HC-SR04, que es compatible con Arduino.

Posteriormente, en la sección de Diseño del programa, se definirá el sistema a implementar considerando su alcance y restricciones. Se mostrará el esquema del circuito que forman la placa Arduino UNO R3 junto al sensor HC-SR04, así como el entorno operacional en el que puede aplicarse.

En Implementación del Programa, se explicará el proceso completo para llevar a cabo la aplicación de medición de distancias, desde la instalación del entorno de desarrollo integrado (IDE) hasta la elaboración del programa y los detalles de comunicación entre la placa Arduino UNO R3, el ordenador con el IDE de Arduino instalado y el sensor ultrasónico HC-SR04..

5.-da Rocha Araújo, H. S., da Silva Filho, D. I. O., Santana, K. A., dos Anjos Oliveira, W., & de Souza, V. O. S. T. (2019). Maquete de semáforo com ferramentas arduino e proteus como método didático de ensino. In *Anais do Congresso Nacional de Educação CONEDU* (Vol. 5, No. 1, pp. 14764-14775).

Descripción: Este artículo propone un sistema de control y monitoreo de carreteras que utiliza métodos de automatización. Este sistema se utilizará como un ejemplo práctico en las aulas para ayudar a los estudiantes a entender mejor estos conceptos. El sistema se programó utilizando Arduino y se montó utilizando Proteus. Nos muestran también el funcionamiento de los 4 semáforos, además de mostrarnos una maqueta del prototipo de los semáforos creados con antelación.

Nos listas los materiales utilizados para la creación de esta misma lo cual ayuda a los lectores a poder implementar el prototipo por cuenta propia.

El sistema, que se programó en Arduino para este artículo, funciona mediante una serie de comandos que determinan el estado de los LEDs en diferentes momentos. Este sistema se diseñó para controlar el tráfico en intersecciones de cuatro vías, por lo que se establecieron reglas específicas para el funcionamiento de los semáforos. Por ejemplo, los semáforos en vías perpendiculares no pueden mostrar la luz verde al mismo tiempo para evitar accidentes.

Además de que no todos los semáforos pueden mostrar la luz roja al mismo tiempo, ya que el objetivo es mantener el tráfico en movimiento.

4. Objetivo Principal:

Este proyecto tiene como objetivo principal investigar la teoría de la computación con un enfoque especial en el uso de autómatas para modelar y regular el funcionamiento de un sistema semafórico.

5. Objetivos Secundarios:

- Implementar un sistema eficiente para el control de semáforos que mejore significativamente el flujo de vehículos y peatones.
- Asegurar la seguridad vial mediante la reducción de la congestión en la intersección.
- Utilizar modelos de autómatas finitos deterministas (DFA) para controlar los ciclos de los semáforos basados en entradas de sensores de ultrasonido.

6. Diseño de estudio

6.1 Descripción del Sistema

Componentes del sistema:

- Control Central (Arduino):
 - Modelo: Arduino Uno
 - Función: Procesar las señales de los sensores de ultrasonido y controlar las luces de los semáforos mediante un modelo de autómata finito determinista(DFA) .
 - Programación: Implementar un DFA que controle el ciclo de los semáforos en función del estado actual y las entradas de los sensores.
- Semáforos Vehiculares:

- Luces: LEDs representando las luces de los semáforos (verde, rojo y ámbar).
- Función: Procesar las señales de los sensores de ultrasonido y controlar las luces de los semáforos mediante un modelo de autómatas finito.
- Control: Reciben comando para cambiar de estado según el modelo de autómata.
- Semáforos Peatonales:
 - Luces: LEDs representando las luces de los semáforos peatonales (verde y rojo).
 - Función: Indicar a los peatones cuando cruzar la calle
 - Control: Reciben comandos del Arduino para cambiar de estado según el modelo de autómata.
- Sistema de Comunicación:
 - Conexión: Cables conectando sensores y LEDs al Arduino.
 -

6.2 Interacción entre componentes:

- Sensores Detectan Presencia:
 - Los sensores de ultrasonido detectan la presencia de vehículos midiendo la distancia y envían señales al arduino.
- Controlador Central Procesa Información:
 - El Arduino recibe señales de los sensores
- Controlador Envía Comandos a los Semáforos:
 - El Arduino cambia las luces de los semáforos según las decisiones tomadas por el DFA.
- Semáforos Indican a Conductores y Peatones:
 - Las luces de los semáforos cambian, regulando el flujo de tráfico y peatones de la intersección.

7. Diagrama de estado y transiciones:

7.1 Diagrama de Transiciones:

El diagrama de transición utiliza un temporizador como mecanismo de transición entre los estados.

1: Indica que ha llegado la señal del temporizador, por lo tanto, se debe realizar una transición al siguiente estado.

0: Indica que el sistema sigue esperando la señal del temporizador, permaneciendo en el estado actual.

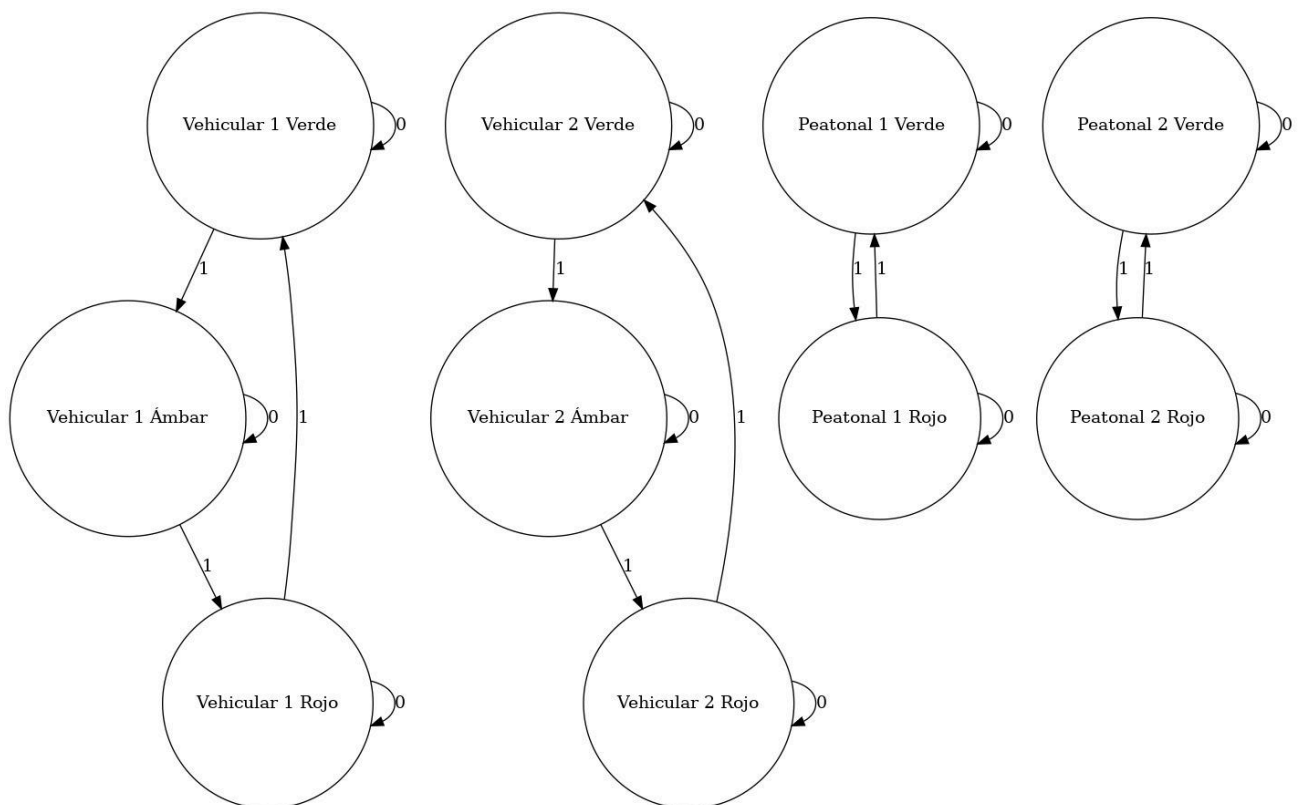
Vehicular 1: Inicia en verde y se mantiene cuando recibe 0, transita al ámbar cuando la señal es 1 y se mantiene cuando es 0, luego transita al rojo cuando recibe 1 y se mantiene en rojo cuando recibe 0 y transita nuevamente al verde inicial cuando recibe 1.

Vehicular 2: Inicia en rojo y se mantiene cuando recibe 0, transita al verde cuando la señal es 1 y se mantiene cuando es 0, luego transita al ámbar cuando recibe 1 y de ahí se mantiene en ámbar cuando recibe 0 y transita nuevamente al rojo inicial cuando recibe 1.

Peatonal 1: Inicia en verde y se mantiene si recibe 0, transita al rojo si recibe 1 y luego se mantiene en rojo si recibe 0, al final si recibe 1 regresa al verde.

Peatonal 2 : Inicia en rojo y se mantiene si recibe 0, transita al verde si recibe 1 y luego se mantiene en verde si recibe 0, al final si recibe 1 regresa al rojo.

Figura 1: Diagrama de transiciones



Nota: Elaboración Propia

7.2 Diagrama de estados:

Podemos definir los estados del AFD de la siguiente manera:

Estado 1: Semáforo 1 en verde, semáforo 2 en rojo, peatonal 1 en rojo, peatonal 2 en verde.

Estado 2: Semáforo 1 en amarillo, semáforo 2 en rojo, peatonal 1 en rojo, peatonal 2 en verde.

Estado 3: Semáforo 1 en rojo, semáforo 2 en verde, peatonal 1 en verde, peatonal 2 en rojo.

Estado 4: Semáforo 1 en rojo, semáforo 2 en amarillo, peatonal 1 en verde, peatonal 2 en rojo.

Las transiciones entre estos estados se realizan en función del tiempo.

Definición del AFD

Estados: {S1, S2, S3, S4}

Estado Inicial: S1

Transiciones:

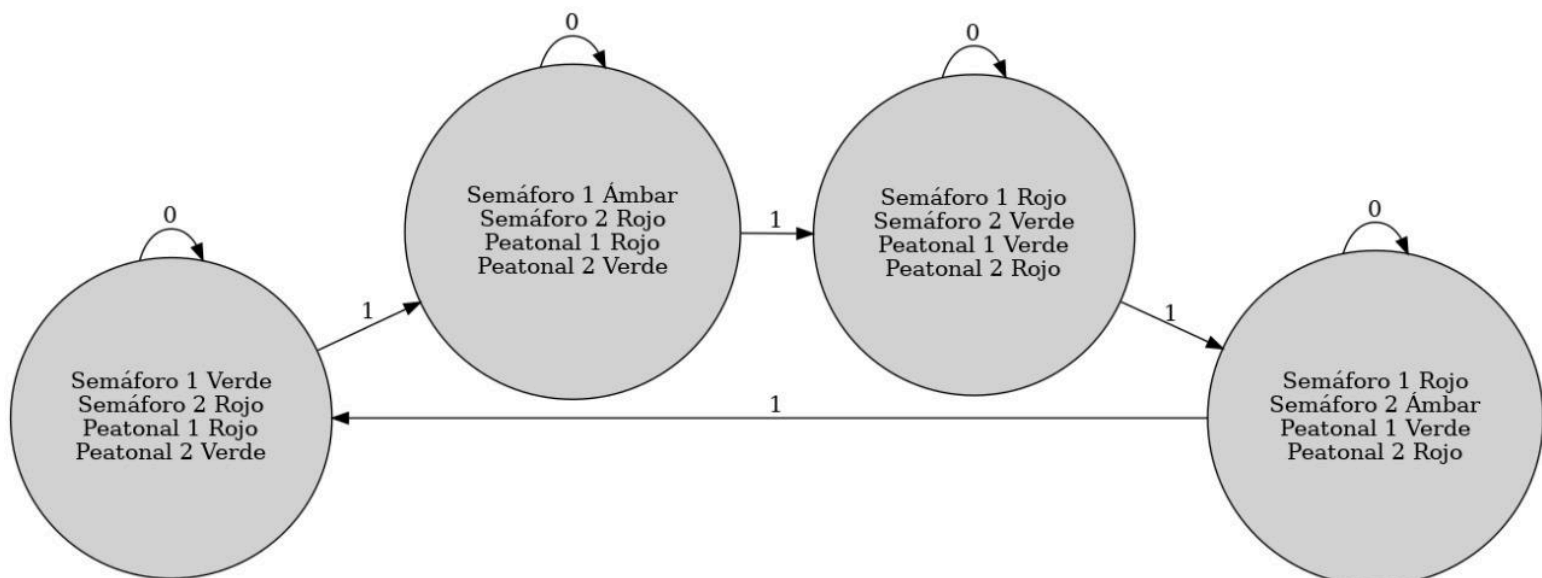
S1 -> S2 (después de 3 segundos)

S2 -> S3 (después de 1 segundo)

S3 -> S4 (después de 3 segundos)

S4 -> S1 (después de 1 segundo)

Figura 2: Diagrama de estados

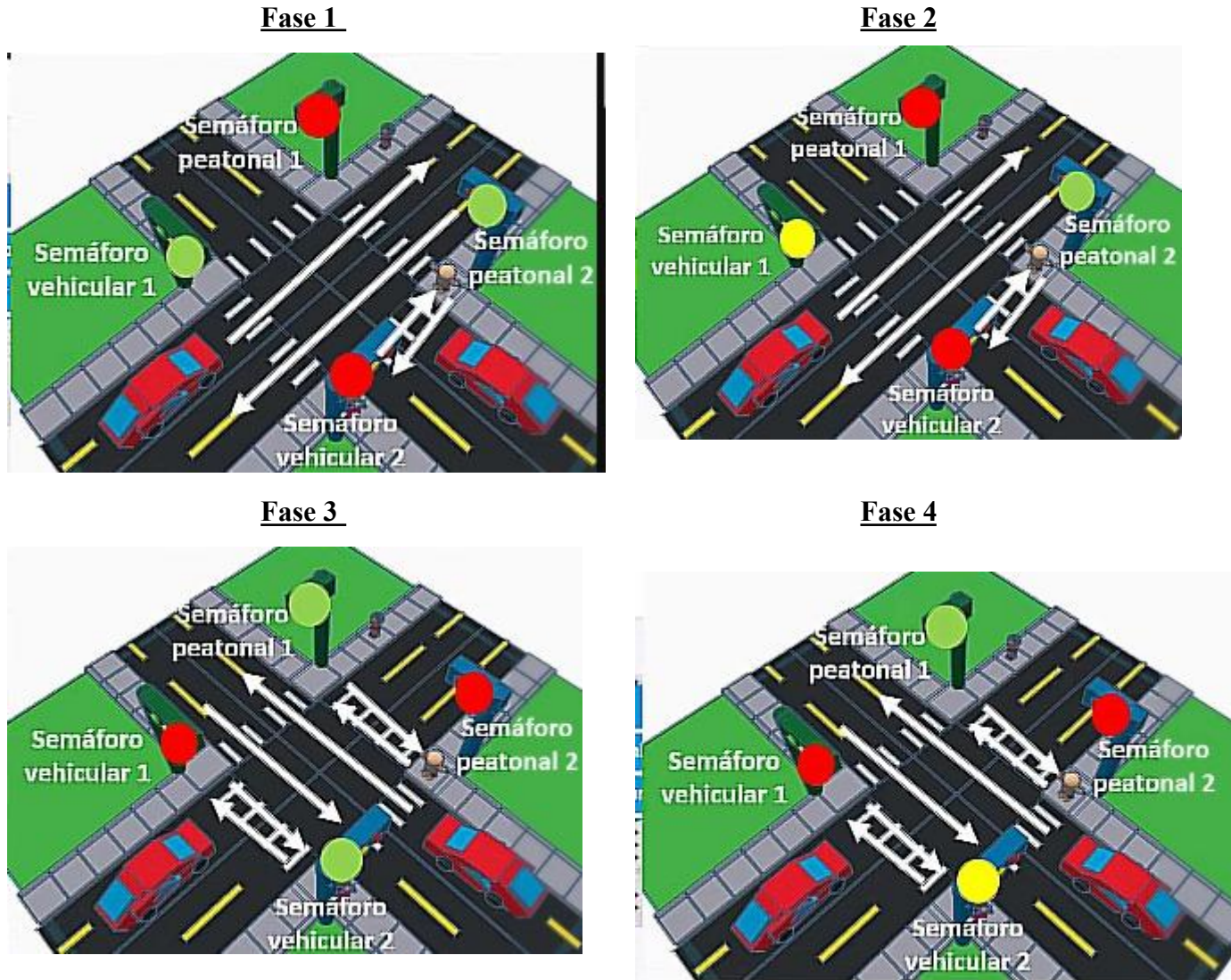


Nota: Elaboración Propia

8. Resultados:

Los resultados esperados los dividimos en 4 fases:

Figura 3: Fases



Nota: Herramienta computacional para sugerir y validar políticas de luces de semáforos.

Obtenido de ResearchGate

Estos resultados incluyen:

- **Mejora en la Gestión del Tráfico:** Un flujo vehicular más eficiente y seguro en la intersección.
- **Reducción de la congestión:** Menos tiempo de espera para conductores y peatones.
- **Seguridad Vial:** Menor riesgo de accidentes debido a una mejor coordinación de los semáforos.

- **Prototipo Funcional:** Un modelo operativo del sistema de control de semáforos utilizando Arduino.

9. Implementación:

En la implementación que realizamos sigue una secuencia de las fases mencionadas anteriormente. En la primera parte del código, se definen constantes para cada uno de los puertos digitales que se utilizarán para controlar las luces. Por ejemplo, **const int L11 = 11;** define una constante llamada **L11** con el valor **11**, que se refiere al puerto digital 11 del microcontrolador.

Definición de estados

Luego, se define nuestros **Estados** que tiene cuatro valores posibles: **ESTADO_1**, **ESTADO_2**, **ESTADO_3** y **ESTADO_4**. Estos estados se utilizarán para controlar el comportamiento de las luces.

Configuración inicial

En la función **setup()**, se configuran los puertos digitales como salidas utilizando la función **pinMode()**. Esto permite que el microcontrolador envíe señales eléctricas a los puertos digitales para controlar las luces.

Bucle principal

La función **loop()** es el bucle principal del programa, que se ejecuta repetidamente. En este bucle, se utiliza un **switch** para determinar en qué estado actual se encuentra el sistema y ejecutar las instrucciones correspondientes.

Estados

Cada caso del **switch** corresponde a un estado diferente. En cada estado, se utilizan las funciones **digitalWrite()** para establecer el estado de cada luz (HIGH o LOW, que corresponden a encendido o apagado, respectivamente). Luego, se utiliza la función **delay()** para esperar un cierto tiempo antes de cambiar al estado siguiente.

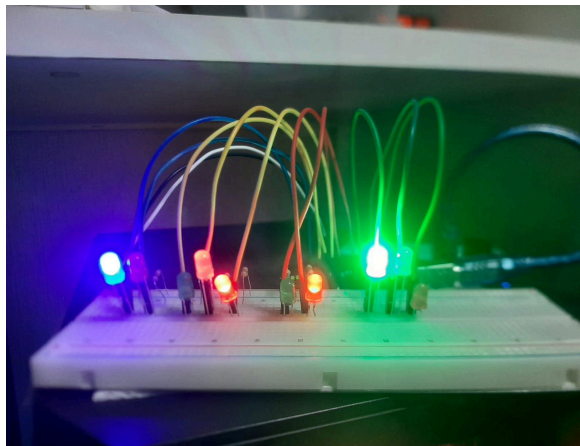
Por ejemplo, en el estado **ESTADO_1**, se encienden las luces conectadas a los puertos L11, L10, L7 y L2, y se apagan las demás. Luego, se espera 4 segundos antes de cambiar al estado **ESTADO_2**.

Transición entre estados

En cada estado, se establece el estado siguiente utilizando la instrucción **estadoActual = <nuevo_estado>;**. Por ejemplo, en el estado **ESTADO_1**, se establece **estadoActual = ESTADO_2;** para cambiar al estado **ESTADO_2** después de esperar 4 segundos.

Fase 1:

Figura 4: Circuito Fase 1

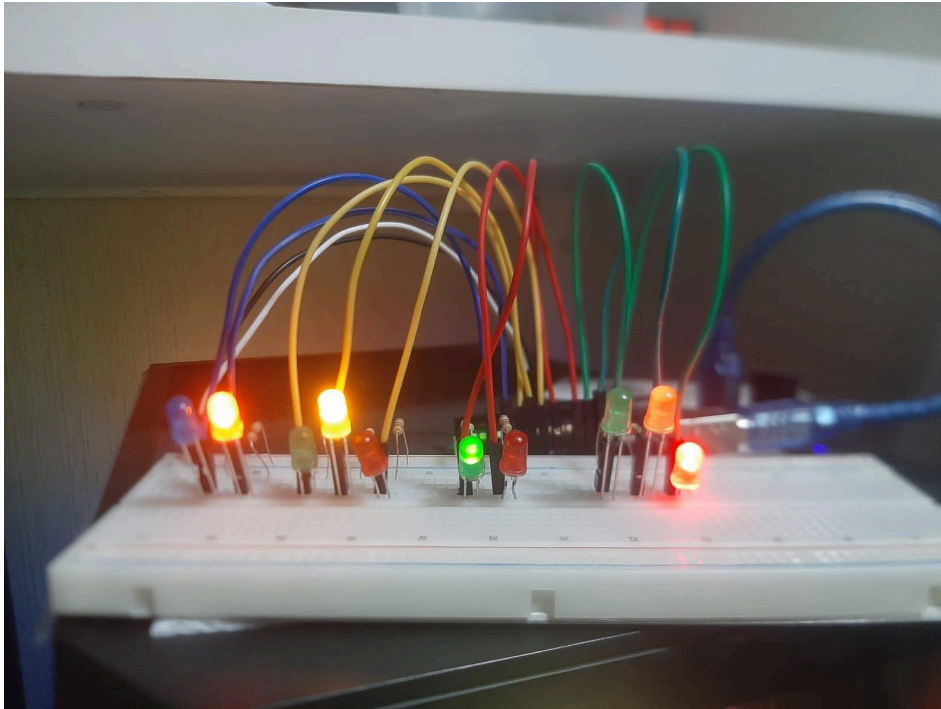


Nota: Elaboración propia: Semáforo con Arduino basado en Autómatas

En esta fase se puede apreciar como nuestro Semáforo vehicular 1 está encendido en luz verde y su Semáforo peatonal 1 en luz roja, lo cual es inverso por el Semáforo vehicular 2 y Semáforo peatonal 2. Esto se debe a nuestro primer estado que define a los LEDs de esta manera siendo nuestro valor de 1 y 0, es decir, los valores para las transiciones el tiempo que se toma estar prendido.

Fase 2:

Figura 5: Circuito Fase 2

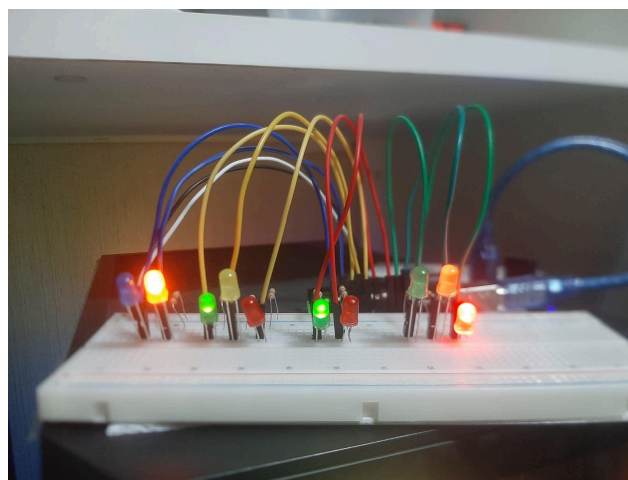


Nota: Elaboración propia: Semáforo con Arduino basado en Autómatas

En esta fase se puede presenciar después de la transición del Estado 1, en el Estado 2 todos los LEDs estados se mantienen todos excepto el Semáforo vehicular 1 lo cual cambia a ámbar para la transición de los peatones y la transición al Estado 3, siendo nuevamente el tiempo nuestro valor para las transiciones.

Fase 3:

Figura 6: Circuito Fase 3

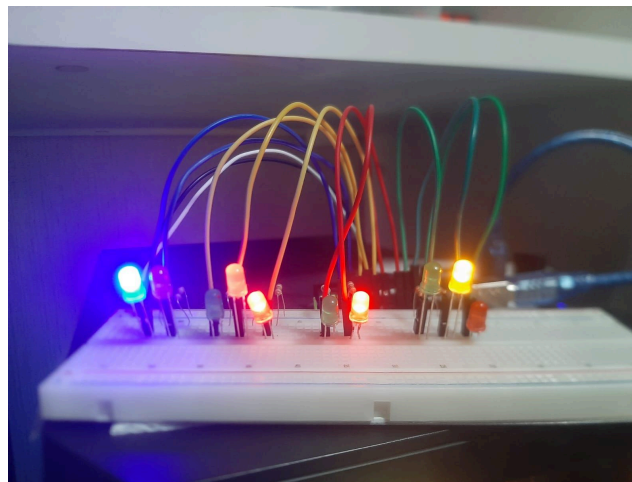


Nota: Elaboración propia: Semáforo con Arduino basado en Autómatas

En esta fase nos encontramos en el Estado 3, siendo aquí donde se ‘invierten’ los papeles, es decir; el Semáforo vehicular 1 se encontrará con el LED rojo encendido y su Semáforo peatonal se encontrará con el LED verde encendido y en el caso del el Semáforo vehicular 2 se encontrará con el el LED verde encendido y su Semáforo peatonal se encontrará con el LED rojo encendido siendo opuestos y nuevamente su valor para la transición al Estado 4 será el tiempo de encendido del LED.

Fase 4:

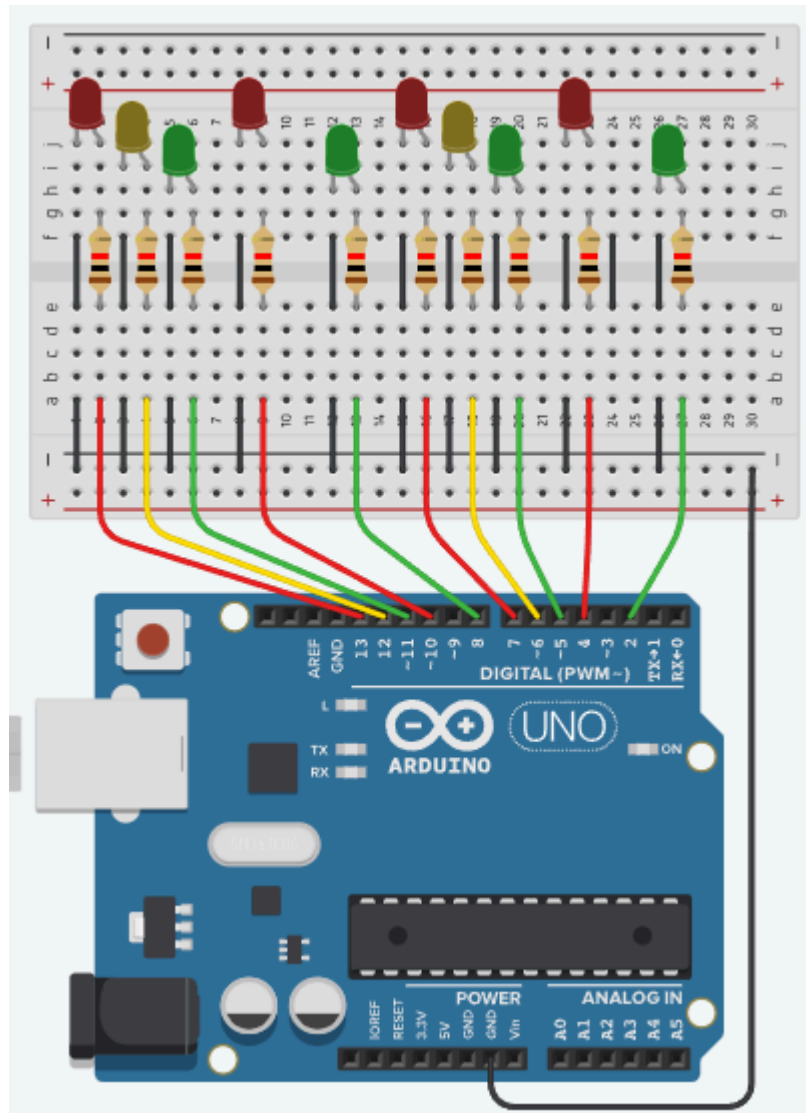
Figura 7: Circuito Fase 4



Nota: Elaboración propia: Semáforo con Arduino basado en Autómatas

En esta fase nos encontramos en el estado final, el Estado 4, este estado funcionara como un bucle, es decir se repetirá el mismo proceso que el Estado 1 con la única diferencia de que el Semáforo Vehicular 2 será el que está en ámbar y los demás se quedan de la misma manera que en el Estado 3. Una vez acabe el tiempo de espera volveremos al Estado 1, ya que estamos simulando cuatro Semáforos en la intersección de cuatro calles, el circuito debe continuar.

Figura 8: Circuito Arduino en Tinkercad




Nota: Elaboración propia. Circuito de Semáforo en Tinkercad.

10. Conclusiones:

En conclusión, utilizando los conocimientos de los autómatas pudimos llegar a implementar un autómata determinista, el cual nos ayudó para poder recrear los estados de un semáforo. En nuestro caso implementamos cuatro semáforos los cuales nos permitieron crear diferentes estados en los cuales cambiaban las luces de los semáforos dependiendo del estado en el que se encontrara actualmente el autómata, además realizando el uso de una arduino, leds y un protoboard pudimos implementar el semáforo con el autómata.

11. Referencias:

- Arduino.cc. (n.d.). Sitio web oficial de Arduino. <https://www.arduino.cc/>
- Isc, E. [@EDUCATRONICOSISC]. (2022, marzo 9).  Control de 4 semáforos usando Arduino y Tinkercad [Video]. Youtube. <https://www.youtube.com/watch?v=VroMQZQvSyw>
- MIT OpenCourseWare. (2020). *The Theory of Computation*. <https://ocw.mit.edu/courses/18-404j-theory-of-computation-fall-2020/>
- ResearchGate. (2015). Estructura de fases de un cruce de 4 calles con 3 fases. https://www.researchgate.net/figure/Estructura-de-fases-de-un-cruce-de-4-calles-con-3-fases-Para-el-estudio-se-considerara_fig1_317264476
- Sandoval, L. [@lesliesandoval9539]. (2021, abril 24). *Semáforo inteligente-Tinkercad* [Video]. Youtube. <https://www.youtube.com/watch?v=vuLVwqfRIW8>
- Teoría de la Computación. (n.d.). ICC - Instituto de Ciencias de la Computación. <https://icc.fcen.uba.ar/teoria-de-la-computacion>

•

12. Anexos

Link del tinkercad:

<https://www.tinkercad.com/things/6YOA66NTNgY-copy-of-arduino-semaforo/edit?sharecode=mmKMFvxpNsw0GP5Uw3sxp0-VLD63zxu0gV3dNYtknWE>

Enlace GitHub:

<https://github.com/ClaudioJL/Trabajo-Final-TC..git>