Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE-0624 Laboratorio de Microcontroladores

Laboratorio #2

Grupo 01

Profesor: MSc. Marco Villalta Fallas

Estudiante: Luis Brenes Campos — **Carné:** C21324
Estudiante: Elizabeth Matamoros — **Carné:** C04652

I SEMESTRE 2025

Índice

1. Introducción

En este laboratorio se desarrolló un sistema de cruce de semáforos simplificado utilizando el microcontrolador ATtiny4313. El objetivo principal fue profundizar en el uso de los GPIOs, tema ya abordado en el laboratorio anterior, y aplicar nuevos conceptos fundamentales como el uso de interrupciones, temporizadores (timers) y el diseño de una máquina de estados finitos (FSM) para el control del flujo lógico del sistema.

En cuanto a los GPIOs estos permitieron interactuar con los periféricos del sistema, en este caso LEDs para representar las luces del semáforo y botones para simular la solicitud de paso peatonal. Se configuraron los registros correspondientes para establecer los pines como entradas o salidas, y se utilizaron resistencias pull-up internas para la correcta detección de los botones.

El control de tiempo se logró mediante el uso de un temporizador en modo CTC (Clear Timer on Compare Match), configurado con un prescaler de 256 y una interrupción cada 0.5 segundos. Estas interrupciones fueron fundamentales para ejecutar acciones sin depender de funciones bloqueantes como delay(), lo que permitió un sistema más eficiente.

Asimismo, para gestionar los eventos en el tiempo, se diseñó una máquina de estados finitos, que regula la secuencia de luces del sistema. Esta FSM cuenta con estados definidos como paso de vehículos, transición con parpadeo, paso de peatones y advertencia de fin de cruce. Las transiciones entre estados dependen de condiciones como el tiempo transcurrido y la interacción del usuario con los botones.

La implementación se realizó en lenguaje C, interactuando directamente con los registros del microcontrolador. La simulación del sistema se llevó a cabo utilizando el software SimulIDE, donde se construyó el circuito con LEDs, botones, resistencias y el ATtiny4313, además de un analizador de señales que permitió observar los cambios de estado en tiempo real y validar el comportamiento del sistema.

El desarrollo completo del laboratorio, que incluye el código fuente, las simulaciones y la documentación del laboratorio, se encuentra disponible en el repositorio oficial del curso en Git UCR, en la carpeta correspondiente al Laboratorio #2.

2. Nota teórica

2.1. Información general del ATtiny4313

El **ATtiny4313** es un microcontrolador AVR de 8 bits desarrollado por Atmel (actualmente parte de Microchip Technology). Este dispositivo pertenece a la familia de microcontroladores *tinyAVR*, los cuales están diseñados para aplicaciones de propósito general que requieren eficiencia, bajo consumo energético y simplicidad en el diseño [1].

Su arquitectura RISC mejorada permite ejecutar la mayoría de las instrucciones en un solo ciclo de reloj, alcanzando hasta 20 MIPS a 20 MHz, lo que ofrece un rendimiento óptimo en aplicaciones donde el tiempo de respuesta es crítico [1].

El ATtiny4313 incluye:

- 4 KB de memoria Flash para almacenamiento de código.
- 256 B de EEPROM para almacenamiento no volátil de datos.
- 256 B de SRAM para almacenamiento de variables y operaciones en tiempo de ejecución.

Además, dispone de 18 líneas de entrada/salida digital (I/O), distribuidas en tres puertos: PORTA, PORTB y PORTD, y múltiples periféricos como:

- Dos temporizadores: uno de 8 bits (Timer/Counter0) y uno de 16 bits (Timer/Counter1).
- Cuatro canales PWM (modulación por ancho de pulso).
- Comunicación serial mediante USART.
- Interfaz USI (Universal Serial Interface) configurable como SPI o I^2C.
- Comparador analógico, watchdog timer y soporte para múltiples fuentes de interrupción.

El microcontrolador también soporta varios modos de bajo consumo (Idle, Power-down, Standby), que permiten reducir significativamente el consumo energético durante periodos de inactividad. Puede operar con un voltaje entre **1.8 V y 5.5 V**, y cuenta con un oscilador interno calibrado de **8 MHz**, además de soportar osciladores externos configurables mediante fuses [1].

Gracias a su tamaño compacto, facilidad de programación en C y conjunto versátil de periféricos, el ATtiny4313 es ideal para aplicaciones como automatización, control de señales, dispositivos embebidos y proyectos académicos.

2.1.1. Diagrama de bloques

A continuación se presenta el diagrama de bloques del microcontrolador **ATtiny4313**, tomado directamente del datasheet oficial proporcionado por Atmel. Este diagrama ilustra los principales módulos funcionales del microcontrolador, incluyendo la unidad de procesamiento central, las memorias, los temporizadores, los puertos de entrada/salida, las interfaces de comunicación y los buses internos que los conectan [2].

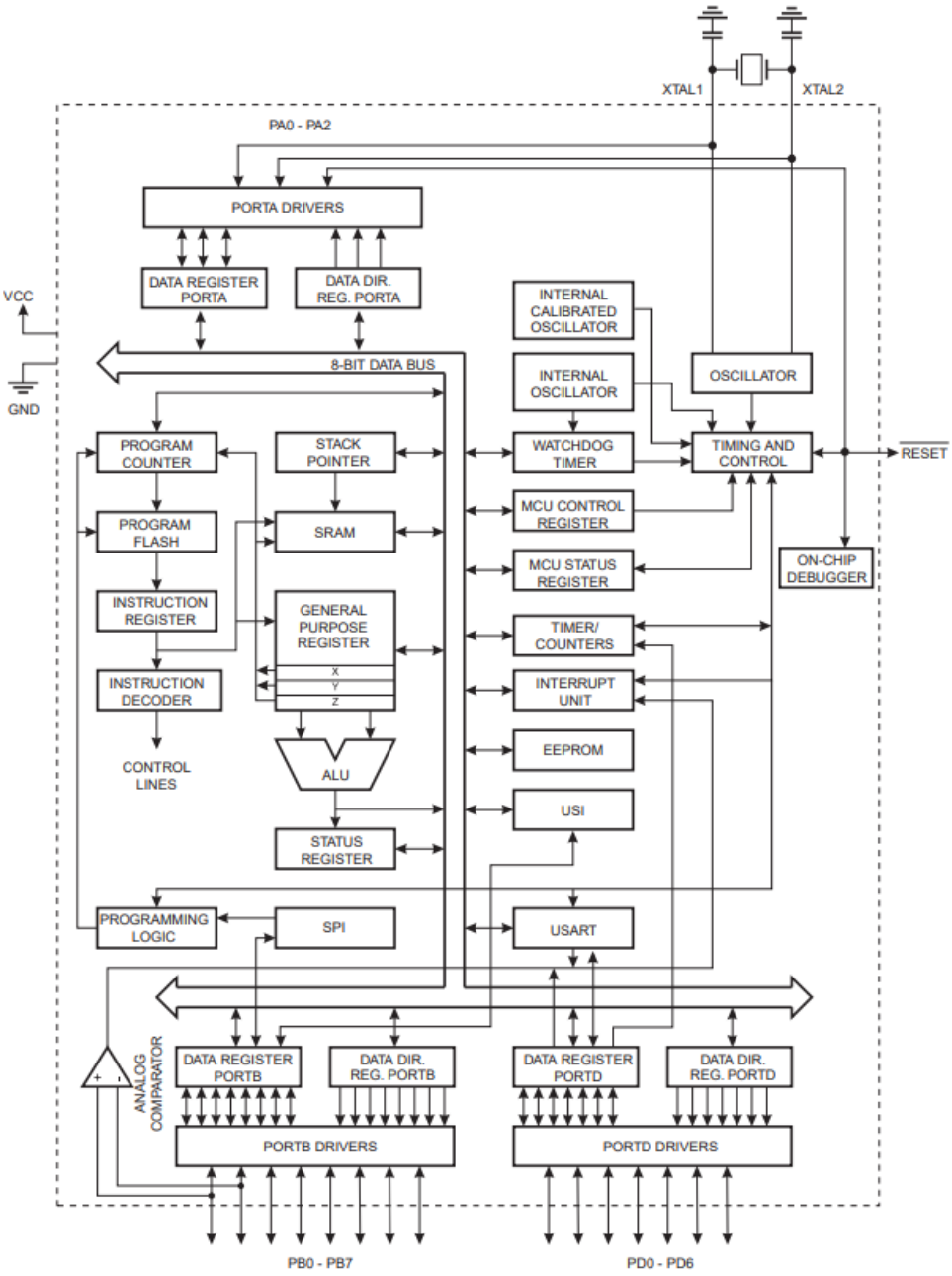

Figura 1: Diagrama de bloque del ATtiny4313 extraído de [2]

2.1.2. Diagrama de pines

En la siguiente figura se muestra el diagrama de pines del microcontrolador **ATtiny4313**, también tomado del datasheet oficial de Atmel. En él se detallan las funciones asignadas a cada uno de los pines físicos del encapsulado PDIP, incluyendo puertos de I/O, líneas de interrupción, interfaces de comunicación, temporizadores y pines de alimentación [2].

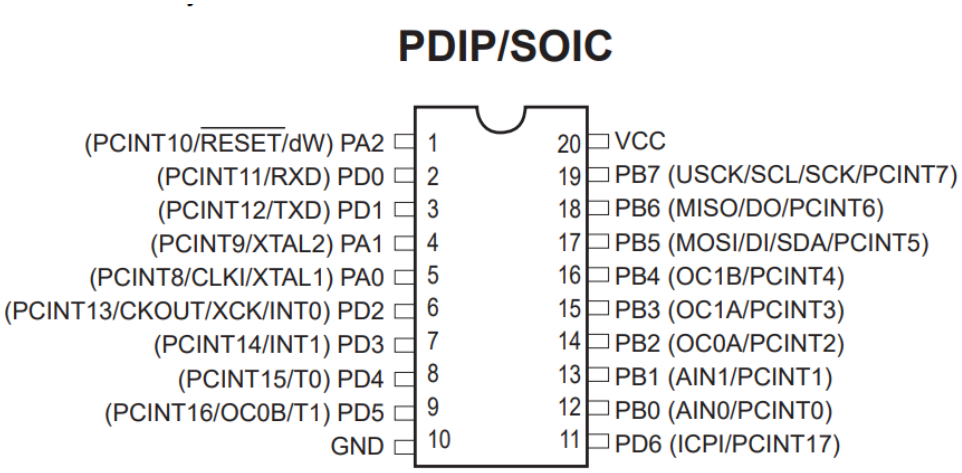

Figura 2: Diagrama de pines del ATtiny4313 extraído de [2]

2.1.3. Especificaciones eléctricas y operativas

El microcontrolador ATtiny4313 presenta características eléctricas que lo hacen adecuado para una amplia gama de aplicaciones en sistemas embebidos, incluyendo ambientes industriales. A continuación, se resumen las principales especificaciones extraídas de su hoja de datos [1]:

- **Líneas I/O programables:** 18 pines digitales configurables como entrada o salida.
- **Voltaje de operación:** entre **1.8 V y 5.5 V**.
- **Grados de velocidad:**
  - 0–4 MHz en el rango de 1.8–5.5 V.
  - 0–10 MHz en el rango de 2.7–5.5 V.
  - 0–20 MHz en el rango de 4.5–5.5 V.
- **Rango de temperatura:** -40 °C a +85 °C (grado industrial).
- **Consumo típico de corriente:**
  - Modo activo: 190 μA a 1.8 V y 1 MHz.
  - Modo inactivo (Idle): 24 μA a 1.8 V y 1 MHz.
  - Modo de apagado (Power-down): 0.1 μA a 1.8 V y 25 °C.
- **Encapsulados disponibles:** PDIP de 20 pines, SOIC y MLF/VQFN de 20 contactos.

2.1.4. Temporización en microcontroladores AVR

La temporización es un aspecto esencial en los microcontroladores, especialmente en aplicaciones donde se requiere una gestión precisa del tiempo, como en la generación de señales, el control de eventos o la ejecución de tareas periódicas. En lugar de utilizar funciones bloqueantes como `delay()`, que detienen la ejecución del programa, el uso de temporizadores permite mantener el flujo continuo del sistema mediante la generación de interrupciones en intervalos definidos.

El microcontrolador ATtiny4313 cuenta con temporizadores internos, específicamente **Timer/Counter0** (de 8 bits) y **Timer/Counter1** (de 16 bits), que pueden operar en distintos modos según la aplicación deseada. Estos temporizadores pueden configurarse para funcionar de forma autónoma o como contadores de eventos externos, aprovechando señales aplicadas a los pines T0 o T1.

Modos de operación del temporizador:

- **Modo 0:** Operación básica, donde el temporizador se incrementa hasta desbordarse (overflow), útil para tareas de conteo continuo.
- **Modos 1 y 3:** Utilizados para la generación de señales PWM (Pulse Width Modulation), permitiendo el control de dispositivos como motores o LEDs con distintos niveles de intensidad.
- **Modo 2 (CTC):** *Clear Timer on Compare Match*, permite que el temporizador se reinicie automáticamente al alcanzar un valor específico, generando interrupciones periódicas sin necesidad de intervención directa del programa.

En el contexto del presente laboratorio, se utilizó el **Timer1** en modo CTC con un prescaler de 256, configurado para generar una interrupción cada 0.5 segundos. Esto permitió controlar el tiempo de cruce del semáforo y la duración de cada estado del sistema sin utilizar retardos bloqueantes. La interrupción asociada a la coincidencia (`TIMER1_COMPA`) permitió actualizar los estados de la FSM de forma precisa y eficiente.

2.2. Periféricos

Durante el desarrollo del laboratorio se utilizaron cuatro periféricos fundamentales del microcontrolador **ATtiny4313**: los GPIOs, la interrupción externa INT0, el temporizador Timer/Counter1 y el sistema global de interrupciones. Estos periféricos fueron configurados directamente desde el código en C, y su uso fue esencial para lograr el comportamiento deseado del sistema de semáforos.

2.2.1. GPIOs (General Purpose Input/Output)

Los GPIOs permiten que el microcontrolador interactúe directamente con su entorno, ya sea mediante la lectura de entradas digitales o el control de salidas como LEDs. El ATtiny4313 cuenta con 18 pines configurables distribuidos en los puertos `PORTA`, `PORTB` y `PORTD` [1].

En este laboratorio, se configuraron como salidas digitales los pines `PB0`–`PB3` para encender y apagar los LEDs del semáforo vehicular y peatonal (verde y rojo). Adicionalmente, se configuró el pin `PD2` como entrada digital para detectar la pulsación de un botón, el cual simula la solicitud de paso peatonal. Este pin se utilizó en conjunto con una interrupción externa.

Los GPIOs se configuraron en función de los registros `DDRx`, `PORTx` y `PINx`, según se describe en la hoja de datos y reforzado en la presentación oficial del curso [3].

2.2.2. Interrupción externa (INT0)

El microcontrolador dispone de una interrupción externa llamada **INT0**, que puede activarse mediante flanco descendente, ascendente o nivel bajo. En este sistema, se utilizó para detectar el momento en que un peatón presiona el botón de cruce conectado a `PD2`. La interrupción se configuró para activarse en flanco descendente, es decir, cuando la señal del botón cambia de alto a bajo, indicando una pulsación.

Esta técnica permite evitar el uso de *polling* (consulta continua del pin), logrando una respuesta inmediata del sistema con bajo consumo de recursos [3].

2.2.3. Temporizador Timer/Counter1

El **Timer1** del ATtiny4313 es un temporizador de 16 bits que permite medir intervalos de tiempo con alta precisión. En este laboratorio se utilizó en modo CTC (Clear Timer on Compare Match) para generar interrupciones periódicas cada 0.5 segundos. Estas interrupciones fueron esenciales para implementar los retardos necesarios en el control del sistema de semáforos, como el tiempo mínimo de paso vehicular o el parpadeo de los LEDs durante las transiciones.

Este uso del temporizador como fuente de interrupciones periódicas es una técnica común y recomendada para aplicaciones con lógica de temporización definida [3].

2.2.4. Sistema global de interrupciones

El sistema global de interrupciones del ATtiny4313 permite habilitar y gestionar todas las fuentes de interrupción disponibles. Para permitir que tanto la interrupción externa como la del temporizador funcionen, se utilizó la

instrucción `sei()` (Set Global Interrupt Enable) en el código principal. Esto habilita el bit I del registro de estado global `SREG`, permitiendo que el microcontrolador ejecute automáticamente las rutinas definidas en el código cuando ocurra una interrupción [1].

2.3. Registros

El control directo del microcontrolador ATtiny4313 en este laboratorio se realizó manipulando registros específicos que permiten configurar los periféricos utilizados, como los GPIOs, el temporizador y las interrupciones. A continuación se describen los registros empleados, clasificados según su función.

2.3.1. Registros para control de E/S digital (GPIO)

- `DDRB`: Se usó para establecer como salidas los pines `PB0`, `PB1`, `PB2` y `PB3`, que controlan los LEDs del sistema de semáforos.
- `PORTB`: Permitió establecer niveles lógicos altos o bajos en los pines de salida para encender o apagar los LEDs vehiculares y peatonales.

2.3.2. Registros para la interrupción externa (INT0)

- `MCUCR`: Se configuró con los bits `ISC01` y `ISC00` para que la interrupción INT0 se dispare por flanco ascendente, correspondiente a la pulsación del botón.
- `GIMSK`: Se activó la interrupción externa INT0 mediante el bit `INT0`.

2.3.3. Registros del temporizador Timer1 (modo CTC)

- `TCCR1B`: Se configuró el temporizador en modo `CTC` mediante el bit `WGM12`, y se seleccionó un prescaler de 256 mediante el bit `CS12`.
- `OCR1A`: Se cargó con el valor 31250 para generar una interrupción cada 0.5 segundos, calculado para un reloj de 8 MHz.
- `TIMSK`: Se habilitó la interrupción por comparación A (OCIE1A), permitiendo que el temporizador dispare la rutina `ISR(TIMER1_COMPA_vect)`.

2.3.4. Registro global de estado

- `SREG`: Registro de estado del microcontrolador. La instrucción `sei()` activa el bit I (Global Interrupt Enable), permitiendo que todas las interrupciones funcionen correctamente.

El uso preciso de estos registros permitió implementar un sistema eficiente, sin necesidad de funciones de alto nivel ni retardos por software. Esta metodología está respaldada tanto por la hoja de datos del microcontrolador como por la información presentada en clase [1, 3].

2.4. Diseño de circuito

El circuito diseñado tiene como objetivo simular el funcionamiento de semáforos peatonales y vehiculares en un cruce pestonal. Ahora bien, en el diseño implementado, los botones de solicitud de paso están conectados al pin `PD2` del ATtiny4313, configurado como entrada digital con interrupción externa (INT0). Al ser presionados, estos botones conectan el pin a nivel alto (+5V). Para asegurar una lectura estable sin componentes adicionales, se utiliza la resistencia *pull-up* interna del microcontrolador, activada mediante el registro `PORTD`.

En cuanto al control de las señales visuales, se utilizaron `LEDs de color rojo y verde` para representar las luces de los semáforos vehicular y peatonal. Los LEDS están conectados a través de resistencias limitadoras de corriente a los pines `PB0{PB3` configurados como salidas digitales. Cada LED tiene una caída de tensión típica de entre 1.8V y 2.2V dependiendo del color, y una corriente de operación recomendada de 10–30mA.

De acuerdo con el datasheet del ATtiny4313, la **corriente máxima absoluta por pin de salida es de 40mA** [1]. Sin embargo, este valor no debe alcanzarse en operación continua, ya que forma parte de los límites máximos absolutos. Para asegurar el funcionamiento confiable del microcontrolador y prolongar la vida útil de los LEDs, se eligió una corriente de diseño de 25mA por LED, que se encuentra dentro del rango seguro de operación.

Aplicando la Ley de Ohm para determinar la resistencia limitadora necesaria, se tiene:

$$R = \frac{V_{DD} - V_{LED}}{I_{LED}} = \frac{5\,\text{V} - 2.0\,\text{V}}{25\,\text{mA}} = 120\,\Omega \tag{1}$$

El valor obtenido fue de 120Ω, el cual corresponde a un valor estándar según la serie E24. Esta elección facilita la implementación práctica, ya que estas resistencias son comúnmente disponibles tanto en laboratorios como en kits de componentes básicos.

Cabe destacar que en el diseño se utilizaron tanto LEDs rojos como verdes. Aunque estos colores presentan ligeras diferencias en la caída de tensión (1.8–2.0V para el rojo y 2.0–2.2V para el verde), la elección de una resistencia común de 120Ω permite mantener la corriente en un rango seguro para ambos tipos. Esta decisión simplifica el diseño del circuito y fue validada mediante simulación, donde no se observaron diferencias funcionales relevantes entre los LEDs utilizados.

Además, el valor fue validado mediante simulación en SimulIDE. Los resultados mostraron que los LEDs se encendían con brillo suficiente sin exceder los límites de corriente del microcontrolador, y el sistema respondió correctamente a las solicitudes de cruce peatonal. No se observaron caídas de voltaje indebidas ni errores de lógica en el sistema, lo que confirmó la adecuación del diseño.

Tabla 1: Parámetros eléctricos considerados en el diseño

Parámetro	**Valor**
Voltaje de alimentación (V_{DD})	5.0V
Caída de voltaje del LED (V_{LED})	2.0V
Corriente de operación por LED (I_{LED})	25mA
Resistencia limitadora utilizada (R)	120Ω
Potencia disipada por la resistencia	0.075W
Potencia disipada por el LED	0.050W

El microcontrolador utilizado fue el ATtiny4313 este presenta una solución compacta y eficiente para aplicaciones de control digital simple. Su disponibilidad en encapsulado PDIP de 20 pines, su facilidad de programación en lenguaje C y su conjunto de periféricos (temporizadores, interrupciones, E/S digitales) lo hacen ideal para implementar la lógica de un sistema de semáforo básico, como el desarrollado en este laboratorio.

Se asumió una fuente de alimentación de 5V, adecuada para el funcionamiento del ATtiny4313 en su rango nominal. En aplicaciones físicas, esta alimentación podría provenir de una fuente regulada o una batería con regulación integrada.

No se incluyeron protecciones adicionales como diodos de protección, capacitores de desacoplo o resistencias de limitación en los botones, dado que se trató de una simulación funcional. En una implementación física, estos componentes serían recomendables para proteger la integridad del microcontrolador y reducir interferencias.

Se incorporó un analizador de señales lógico en la simulación para verificar el comportamiento de las señales digitales en tiempo real. Esto permitió observar con precisión los cambios de estado en los LEDs y la respuesta del sistema ante la activación del botón, validando la correcta implementación de la máquina de estados y la temporización.

El diseño implementado fue el siguiente:

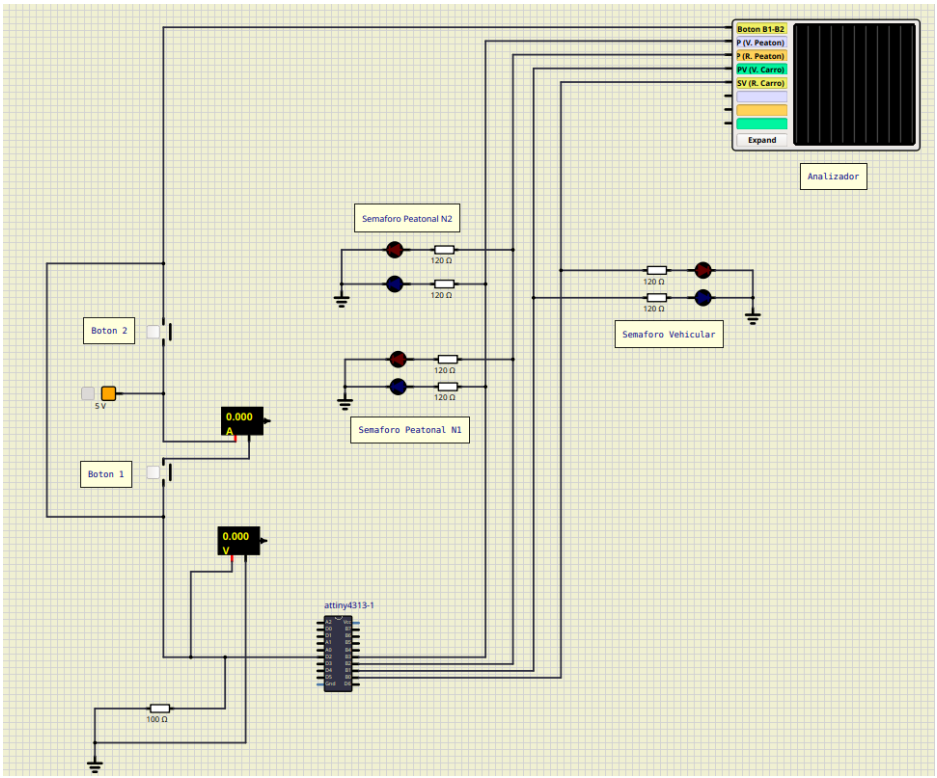

Figura 3: Circuito diseñado

2.5. Lista de componentes y precios

Tabla 2: Lista de componentes y precios estimados en Costa Rica

Componente	Descripción	Precio unitario (CRC)
ATtiny4313-PU	Microcontrolador AVR de 8 bits	₡1 200
LED rojo 5mm	Diodo emisor de luz rojo de 5mm	₡100
LED verde 5mm	Diodo emisor de luz verde de 5mm	₡100
Resistencia 120 Ω	Resistencia de película metálica 120 Ω, 0.25W	₡50
Botón pulsador	Interruptor momentáneo tipo push-button	₡300

Los precios presentados en la Tabla 2 corresponden a estimaciones obtenidas de proveedores locales en Costa Rica, como Micro JPM [4], Teltron [5] y Electro CR [6]. Estos valores son aproximados. Se seleccionaron componentes ampliamente utilizados en proyectos de electrónica básica, priorizando la compatibilidad con el microcontrolador ATtiny4313 y la facilidad de montaje en protoboard. Todos los artículos mencionados están disponibles en formatos estándar, lo que favorece su adquisición en tiendas locales o laboratorios universitarios. Para una implementación física definitiva, se recomienda consultar directamente con los distribuidores locales para comparar precios y confirmar la disponibilidad en el mercado nacional.

2.6. Conceptos/temas del laboratorio

Este laboratorio permitió aplicar y reforzar varios conceptos fundamentales del desarrollo con microcontroladores, específicamente con el **ATtiny4313**. A continuación, se describen los temas principales abordados durante su implementación.

2.6.1. GPIOs

Los pines de propósito general (*General Purpose Input/Output*) permiten al microcontrolador interactuar con su entorno mediante señales digitales. En este proyecto, se configuraron pines como salidas para controlar los LEDs del sistema de semáforos y como entrada para leer el estado de un botón conectado al pin `PD2`. La correcta configuración de los registros `DDRx`, `PORTx` y `PINx` fue esencial para lograr esta funcionalidad [1, 3].

2.6.2. Interrupciones

Las interrupciones permiten que el microcontrolador responda a eventos sin necesidad de supervisión constante (*polling*), suspendiendo temporalmente el flujo principal para ejecutar una rutina específica. En este laboratorio se utilizaron dos tipos de interrupciones:

- **INT0**: interrupción externa activada por flanco ascendente en el pin `PD2`, utilizada para detectar la solicitud de cruce peatonal.
- **TIMER1_COMPA**: interrupción por comparación del temporizador, generada periódicamente para controlar los tiempos del sistema.

Ambas interrupciones fueron gestionadas mediante rutinas específicas (`ISR(...)`), lo que permitió una operación eficiente, sin retardos activos ni pérdida de sincronización.

2.6.3. Temporizador Timer/Counter1

El uso de temporizadores permite medir intervalos de tiempo sin detener la ejecución del resto del programa. En este caso, se utilizó el **Timer1** en modo `CTC (Clear Timer on Compare Match)`, generando una interrupción cada 0.5 segundos. Esta temporización fue fundamental para definir la duración de cada fase del semáforo y permitir el parpadeo de los LEDs durante las transiciones.

2.6.4. Máquina de Estados Finitos (FSM)

El comportamiento del sistema fue modelado como una **máquina de estados finitos**, donde cada estado representa una fase específica del semáforo (paso vehicular, cruce peatonal, parpadeo, etc.). Las transiciones entre estados dependieron de eventos externos (como la pulsación del botón) y del tiempo controlado por el temporizador. En este caso, la FSM se implementó de manera implícita utilizando variables tipo bandera (`flags`), aunque el enfoque puede generalizarse a estructuras más formales utilizando `switch-case` y `enum`, como se recomienda en el curso.

2.6.5. Sistema global de interrupciones

Para que las interrupciones funcionen correctamente, es necesario habilitarlas globalmente mediante la instrucción `sei()`. Esta instrucción activa el bit I del registro `SREG`, permitiendo que el microcontrolador ejecute las rutinas de interrupción cuando se disparan los eventos correspondientes.

2.6.6. Programación a bajo nivel

Todo el desarrollo se realizó directamente sobre los registros del microcontrolador y utilizando rutinas de interrupción escritas en lenguaje C. No se utilizaron bibliotecas de abstracción, lo que permitió un entendimiento más profundo del funcionamiento interno del ATtiny4313 y mayor control sobre los periféricos utilizados. Esta práctica está alineada con los objetivos del curso y con las recomendaciones metodológicas presentadas por el profesor [3].

3. Desarrollo/análisis

3.1. Análisis programa

El programa desarrollado implementa un sistema de control para un semáforo peatonal y vehicular, utilizando el microcontrolador **ATtiny4313**. La lógica se basa en interrupciones externas, temporización mediante el Timer1, y control de señales a través de los pines digitales del puerto `PORTB`.

3.1.1. Diagrama de estas dos

El comportamiento del sistema fue modelado como una **máquina de estados finitos** (FSM), donde cada estado representa una fase del semáforo (paso vehicular, cruce peatonal, parpadeo, etc.) y las transiciones dependen de eventos como la pulsación del botón o el paso del tiempo.

A continuación, se presenta el diagrama de estados que resume el flujo lógico del sistema:

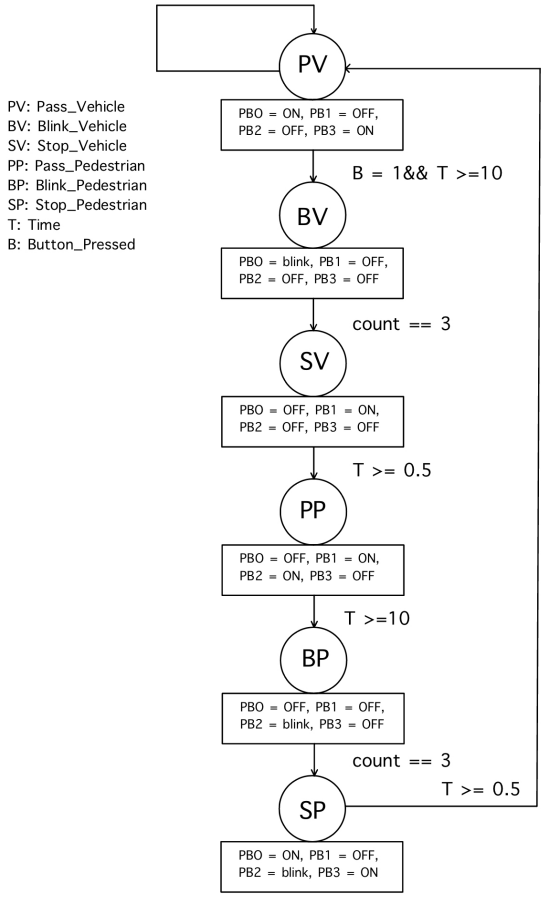

Figura 4: Diagrama de estados

3.1.2. Variables globales

Se definen cuatro variables globales para manejar el estado del sistema:

- `count`: contador general que se incrementa cada 0.5 segundos, utilizado para controlar los tiempos de transición.
- `pedestrian_request`: flag que indica si un peatón ha solicitado cruzar.
- `pedestrian_crossing`: flag que señala si el cruce peatonal está en proceso.
- `pedestrian_end`: flag que indica la transición final del cruce peatonal y la restauración del paso vehicular.

3.1.3. Configuración de periféricos

- `setupPins()`: configura los pines `PB0` a `PB3` como salidas digitales para controlar los LEDs del sistema.
- `setupInterrupt()`: configura el pin `PD2` como entrada con interrupción externa INT0 en flanco ascendente.
- `setupTimer()`: inicializa el Timer1 en modo CTC con un prescaler de 256 y un valor de comparación de 31250, lo que genera una interrupción cada 0.5 segundos con reloj de 16MHz.

3.1.4. Rutinas de interrupción

- `ISR(INT0_vect)`: se activa cuando el botón de solicitud es presionado; establece el flag `pedestrian_request` en alto.
- `ISR(TIMER1_COMPA_vect)`: se ejecuta cada 0.5 segundos. Incrementa el contador `count` y evalúa cuatro funciones que gestionan las transiciones del semáforo:
  - `initialPedestrianCrossing()`: inicia el cruce peatonal si han pasado al menos 10 segundos y hay una solicitud activa.
  - `flikerVehicleLED()`: hace parpadear la luz verde vehicular y luego enciende la roja cuando inicia el cruce peatonal.
  - `endPedestrianCrossing()`: termina el cruce peatonal a los 10 segundos y activa la transición hacia paso vehicular.
  - `flikerPedestrianLED()`: hace parpadear la luz verde peatonal antes de restaurar el estado inicial del sistema.

3.1.5. Lógica del sistema

El sistema opera como una máquina de estados controlada por flags. Inicialmente, los vehículos tienen paso (PB0 encendido), y los peatones deben solicitar cruzar presionando el botón. Si se cumple el tiempo mínimo, el sistema cambia de estado:

1. Parpadea la luz verde vehicular por 3 segundos.
2. Un sengundo después, enciende la luz roja vehicular y la verde peatonal.
3. Después de 10 segundos, parpadea la luz verde peatonal por 3 segundos.
4. Un segundo después, se restablece el paso vehicular y se reinician los estados.

3.1.6. Consideraciones generales

El sistema fue diseñado para ser no bloqueante y eficiente, utilizando temporización por interrupciones en lugar de retardos activos (`delay()`). Las decisiones de transición se basan únicamente en el valor del contador y los flags, asegurando así un comportamiento determinista y confiable. Esta implementación también permite futuras extensiones, como ajustes de tiempo dinámico o múltiples botones de cruce.

3.2. Análisis electrónico

Para verificar el funcionamiento del sistema de semáforos implementado, se realizaron simulaciones en el entorno **SimulIDE**, utilizando el microcontrolador `ATtiny4313` y representando los semáforos con **LEDs conectados a través de resistencias limitadoras de 120 Ω**. En las siguientes figuras se muestran tres momentos clave del funcionamiento, los cuales permiten validar la lógica del control y el comportamiento eléctrico esperado.

Estado inicial (Figura 5): El sistema se encuentra en estado de reposo, con la luz verde del semáforo vehicular encendida (`PB0`) y la luz roja peatonal activa (`PB3`). Este es el estado por defecto del sistema cuando no se ha solicitado cruce peatonal.

Transición de cruce peatonal (Figura 6): Una vez presionado el botón de cruce (conectado a `PD2`), el sistema activa la rutina de interrupción externa (`INT0`). Tras cumplirse el tiempo mínimo de espera de 10 segundos (20 ciclos de 0.5 s), la luz verde vehicular comienza a parpadear, se apaga, y se enciende la luz roja vehicular (`PB1`) junto con la luz verde peatonal (`PB2`). Esto habilita el cruce seguro para peatones.

Restauración del paso vehicular (Figura 7): Luego de 14 segundos desde el inicio del cruce peatonal, la luz verde peatonal parpadea como advertencia, se apaga, y se restablece el estado inicial: la luz verde vehicular vuelve a encenderse, y las señales peatonales se apagan. El sistema reinicia internamente el contador y los indicadores de estado, quedando listo para un nuevo ciclo.

Durante toda la simulación, el **analizador lógico digital** incorporado permitió observar la activación y desactivación de cada señal de salida. Esto validó no solo la lógica funcional, sino también que las salidas del microcontrolador respetan los niveles de voltaje y tiempos definidos. Las resistencias de 120 Ω aseguraron un consumo de corriente seguro en cada LED (aproximadamente 25 mA), respetando los límites del microcontrolador y evitando sobrecarga.

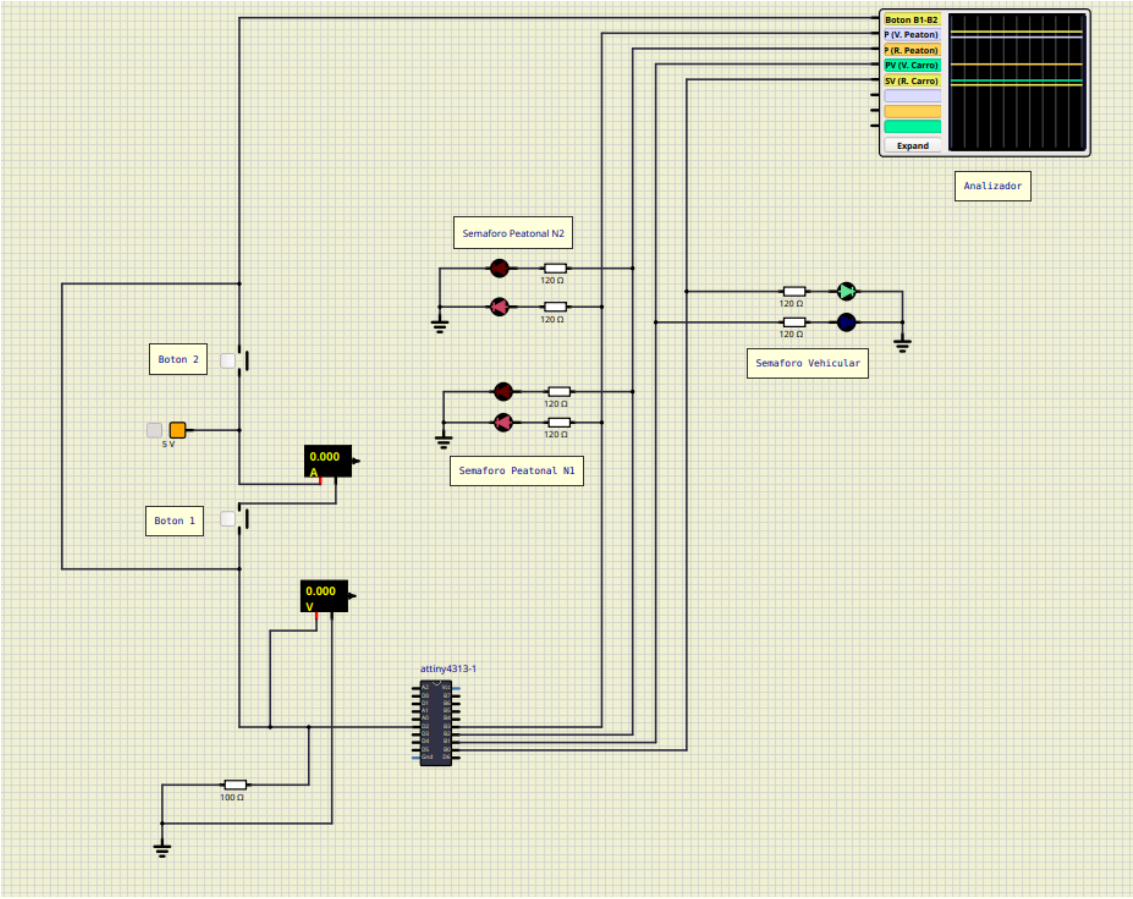

Figura 5: Estado inicial del sistema: paso vehicular habilitado.

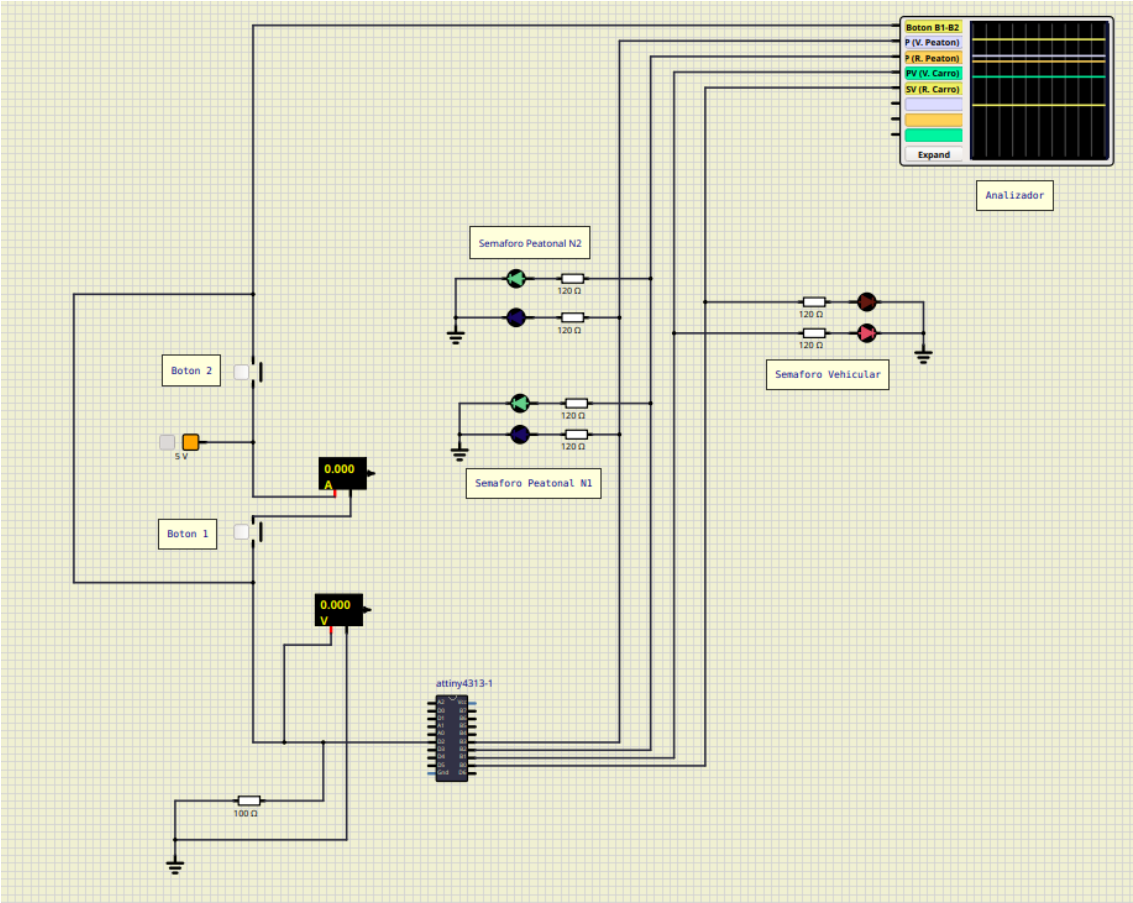

Figura 6: Cruce peatonal activo: luz verde peatonal encendida.

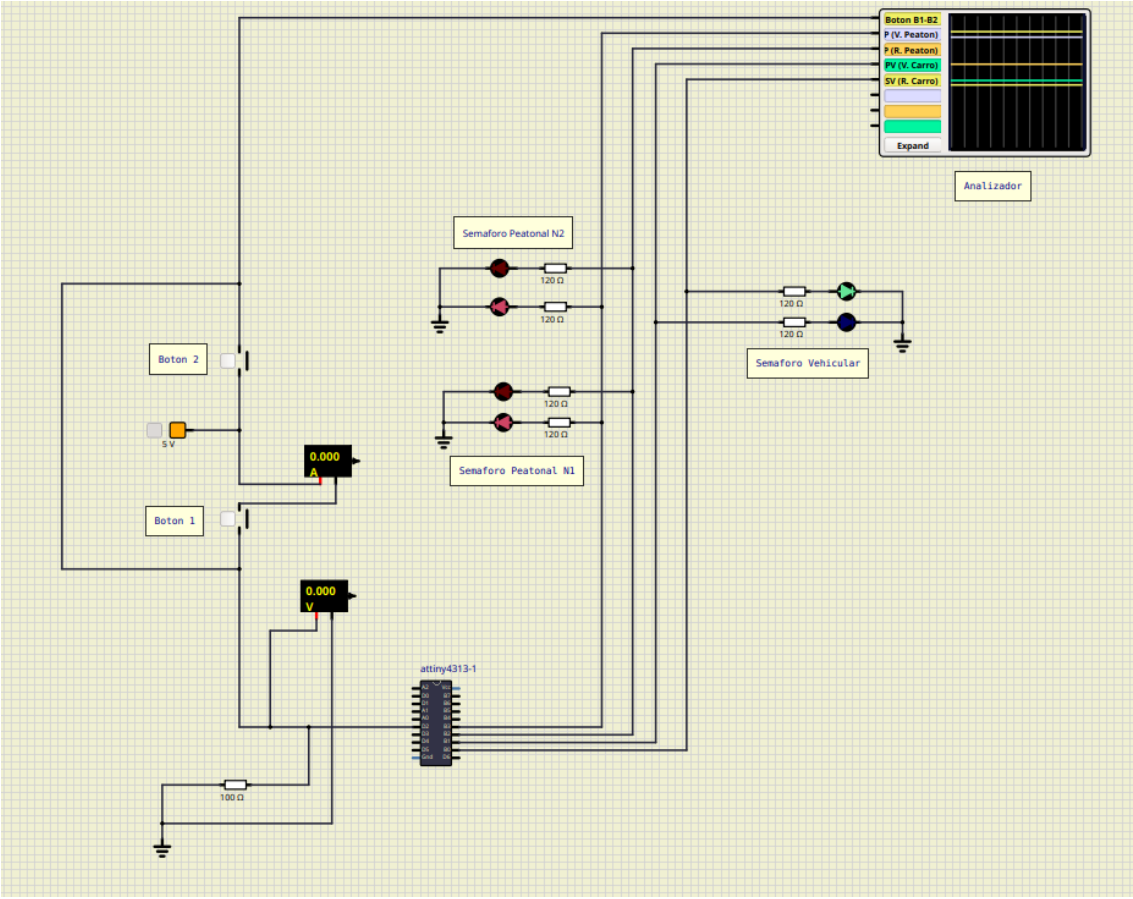

Figura 7: Restauración del paso vehicular tras finalizar el cruce.

4. Conclusiones y recomendaciones

4.1. Conclusiones

- Se logró implementar exitosamente un sistema de semáforo peatonal-vehicular utilizando el microcontrolador ATtiny4313, aplicando técnicas de programación a bajo nivel y control directo mediante registros.
- El uso del temporizador Timer1 en modo CTC permitió generar interrupciones periódicas de forma precisa sin recurrir a retardos activos (`delay()`), lo cual resultó fundamental para el control de tiempos y la eficiencia del sistema.
- La interrupción externa INT0 permitió una detección inmediata de la solicitud de paso peatonal mediante flanco ascendente en el pin `PD2`, eliminando la necesidad de monitoreo constante (polling) y reduciendo el consumo de recursos.
- A través de la simulación en SimulIDE, se comprobó el correcto comportamiento del sistema en sus distintos estados. Se validó la secuencia lógica esperada en cada transición de luces, así como el cumplimiento de los tiempos definidos.
- El desarrollo del laboratorio permitió integrar conceptos clave como **GPIOs**, interrupciones, temporizadores y máquinas de estados finitos, fortaleciendo la comprensión del control embebido en sistemas digitales.

4.2. Recomendaciones

- Para una implementación física, se recomienda incluir protecciones adicionales, como diodos de protección, capacitores de desacoplo y resistencias en serie con los botones, con el fin de evitar rebotes o daños por transitorios eléctricos.
- Sería conveniente optimizar el consumo energético utilizando los modos de bajo consumo del ATtiny4313 (por ejemplo, `Idle` o `Power-down`) cuando el sistema esté en estado de espera.
- Implementar una rutina de anti-rebote por software o mediante hardware con un filtro RC para mejorar la confiabilidad de la detección de los botones, especialmente en ambientes reales donde los rebotes son comunes.
- Evaluar el comportamiento del sistema frente a fallos (por ejemplo, botón atascado o LED dañado) e implementar mecanismos básicos de tolerancia a fallos o reinicio del sistema con ayuda del `Watchdog Timer`.

Referencias

[1] Microchip Technology Inc. *ATtiny2313A/4313: 8-bit AVR Microcontrollers with 2/4K Bytes In-System Programmable Flash*. Rev. 8246B–AVR–09/11. Disponible en: `https://ww1.microchip.com/downloads/en/DeviceDoc/doc8246.pdf`

[2] Atmel Corporation (ahora parte de Microchip Technology Inc.). *ATtiny2313A/4313: 8-bit AVR Microcontrollers with 2/4K Bytes In-System Programmable Flash*. Rev. 8246B–AVR–09/11. Disponible en: `https://ww1.microchip.com/downloads/en/DeviceDoc/doc8246.pdf`

[3] M. Villalta Fallas. *GPIO, Interrupciones, Timers y el ATtiny4313*. Presentación oficial del curso IE-0624 Laboratorio de Microcontroladores, Universidad de Costa Rica, 2025.

[4] Micro JPM, "Nosotros", [En línea]. Disponible en: `https://www.microjpm.com/nosotros/`. [Accedido: 11-abr-2025].

[5] Teltron Costa Rica, "Inicio", [En línea]. Disponible en: `https://teltroncr.com/`. [Accedido: 11-abr-2025].

[6] Electro CR, "Inicio", [En línea]. Disponible en: `https://electrocr.tech/`. [Accedido: 11-abr-2025].