

Proyecto:

Controlador de semáforos con predicción de enventos.

Jafet Soto Arrieta, B77543
Universidad de Costa Rica, Escuela de Ingeniería Eléctrica
IE-0624 Laboratorio de Microcontroladores
3 de junio del 2024
jafet.soto@ucr.ac.cr

Resumen—

Este proyecto desarrolla un controlador de semáforos y una red de LEDs para simular el flujo vehicular en cuatro intersecciones de dos calles con dos carriles cada una. Se enfoca en controlar los semáforos durante los estados de verde y rojo, excluyendo giros a la izquierda para simplificar la complejidad. Los LEDs se agrupan y controlan eficientemente utilizando recursos limitados del Arduino. Se utiliza una pantalla LCD 20x4 para visualizar el estado del controlador y mejorar la presentación de datos. El objetivo final es implementar un algoritmo predictivo ligero que funcione en el Arduino Uno para optimizar el control del tráfico en un entorno simulado de intersecciones urbanas.

I. INTRODUCCIÓN

Este proyecto tiene como objetivo diseñar y desarrollar un controlador de semáforos inteligente y una red de LEDs para simular y gestionar el flujo vehicular en intersecciones urbanas complejas. El sistema se enfoca en optimizar el control del tráfico utilizando recursos limitados disponibles en el Arduino Uno. Se ha simplificado la complejidad del sistema al concentrarse en el control de los semáforos durante los estados de verde y rojo, excluyendo giros a la izquierda para mejorar la eficiencia operativa y la implementación. Además, se ha integrado una pantalla LCD 20x4 para ofrecer una visualización clara y detallada del estado del controlador y los datos del tráfico.

La construcción del hardware implica la integración cuidadosa de LEDs agrupados en pares rojos y verdes, controlados eficientemente por salidas específicas del Arduino Uno debido a limitaciones de puertos. Esto permite una gestión efectiva de los recursos hardware disponibles, asegurando un funcionamiento estable y confiable del sistema de semáforos simulado. Se han combinado LEDs amarillos con verdes y rojos, junto con un módulo RGB, para completar el diseño de los 16 semáforos necesarios.

El desarrollo del código de control se centra en la programación del microcontrolador (MCU) para gestionar los estados de los LEDs y la comunicación con la pantalla LCD. Se ha optimizado el código para garantizar tiempos de respuesta rápidos y precisos durante los cambios de estado de los semáforos, manteniendo al mismo tiempo una estructura modular que facilita futuras expansiones y mejoras.

La implementación de un algoritmo predictivo en el sistema es crucial para anticipar y gestionar el flujo de vehículos de manera eficiente. Se busca integrar un algoritmo ligero y eficaz que pueda ejecutarse en el Arduino Uno, utilizando datos históricos o sensores para predecir patrones de tráfico y ajustar dinámicamente los ciclos de semáforo para minimizar congestiones y tiempos de espera.

II. OBJETIVO GENERAL

Implementar un controlador de tráfico vehicular para un sistema de 4 cruces que permita evitar la congestión vial.

II-A. *Objetivos específicos:*

1. Crear un sistema que simule los 16 semáforos de los 4 cruces vehiculares.
2. Detectar y prevenir la congestión vehicular según los cambios en el sistema.
3. Implementar un algoritmo de detección de patrones para las calles del sistema (HoltWinters).

III. ALCANCE

El sistema desarrollado en este proyecto es una simulación de 16 semáforos distribuidos en cuatro cruces. Cada intersección está compuesta por dos calles con dos carriles cada una, lo que genera un total de cuatro cruces. La idea principal es diseñar y programar un microcontrolador, utilizando un Arduino Uno R3, que sea capaz de manejar el tráfico de manera eficiente en estas cuatro intersecciones.

El sistema está diseñado para controlar los estados de los semáforos (verde y rojo) de manera coordinada, evitando giros a la izquierda para simplificar la complejidad del control y mejorar la eficiencia. Además, se utilizará un algoritmo de control ligero para anticipar y gestionar el flujo vehicular en función de datos en tiempo real.

Se implementará una pantalla LCD 20x4 para visualizar los estados de la máquina de estados y los resultados de las pruebas de simulación, proporcionando una interfaz visual que facilita el monitoreo y la depuración del sistema.

Este proyecto abarca tanto el desarrollo del hardware, incluyendo la configuración y conexión de los LEDs y otros componentes, como el desarrollo del software necesario para el control de los semáforos y la visualización de datos. La meta

IV. JUSTIFICACIÓN

vehicular afecta la mov

Además, la implementación de un sistema de control de tráfico inteligente en un entorno urbano puede reducir significativamente la probabilidad de accidentes, al gestionar de manera óptima los tiempos de espera y el flujo de vehículos. Esto tiene un impacto positivo en la seguridad vial, protegiendo tanto a conductores como a peatones.

V. MARCO TEÓRICO

V-A. *Arduino UNO R3:*

El Arduino Uno R3 está equipado con el microcontrolador ATmega328P y opera a un voltaje de 5V, con un rango recomendado de entrada de 7 a 12V. Puede ser alimentado tanto a través del puerto USB tipo B como mediante un conector de barril de 2.1mm. Este microcontrolador ofrece 14 pines digitales de entrada/salida, de los cuales 6 pueden generar señales PWM. También cuenta con 6 pines de entrada analógica y soporte para programación ISP a través de los puertos TX/RX. Los niveles lógicos de las E/S digitales son de 5V.

The diagram shows a blue PCB with various components and pin headers. The components include a microcontroller, several integrated circuits, and a large black component. The pin headers are labeled with pin numbers and functions. A legend on the left identifies pin types: Control (blue), GND (black), Analog Pin (green), Physical Pin (grey), Port Pin (pink), Serial Pin (yellow), Interrupt Pin (orange), Pin Function (purple), and INT (light green). A red button is located at the top right. The board is labeled 'PCBoard.ca' in the center.

Legend:

- Control
- GND
- Analog Pin
- Physical Pin
- Port Pin
- Serial Pin
- Interrupt Pin
- Pin Function
- INT





Pin Headers:

- IC: 0, 5V, RES, 5V, 5V, 5V, GND, GND, VIN
- POWER: 3.3V, 3.1V, GND, VIN
- ANALOG IN: A0, A1, A2, A3, A4, A5
- DIGITAL: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742,

V-B. Pantalla LCD:

Para integrar la pantalla LCD en el proyecto, se utiliza la biblioteca `LiquidCrystal_I2C` [1], que facilita la comunicación con la pantalla a través del protocolo I2C. Cabe destacar que esta biblioteca no se encuentra incluida por defecto en el IDE de Arduino, por lo que debe descargarse e instalarse manualmente desde el gestor de bibliotecas de Arduino.

LCD 20x4 I2C uses I2C interface, so it has 4 pins:

-  **GND pin:** needs to be connected to **GND** (0V).
-  **VCC pin:** the power supply for the LCD, needs to be connected to **VCC** (5V).
-  **SDA pin:** I2C data signal
-  **SCL pin:** I2C clock signal

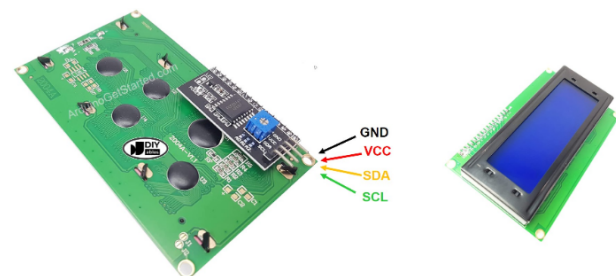


Figura 2: Configuración de pines de la pantalla LCD 20x4.

Esta implementación proporciona una base sólida para el monitoreo y la depuración del sistema, facilitando el proceso de desarrollo y asegurando que el controlador de semáforos funcione de manera óptima.

V-C. *Modulo RGB:*

Para completar el sistema de semáforos y alcanzar un total de 16 semáforos, se ha utilizado un módulo RGB. Este módulo proporciona una solución eficaz para cubrir la falta de un par de LEDs necesarios. El módulo RGB permite controlar las luces roja, verde y azul de manera independiente, pero en este proyecto solo se utiliza la luz verde.

La luz verde del módulo RGB está controlada por el puerto 3 del Arduino Uno R3. Esto se debe a la necesidad de controlar un semáforo adicional sin exceder la cantidad de pines disponibles en el microcontrolador. La integración del módulo RGB en el sistema permite mantener la uniformidad en el control de los semáforos, asegurando que todos los cruces operen de manera sincronizada.

A continuación, se presenta una imagen del módulo RGB utilizado en el proyecto:

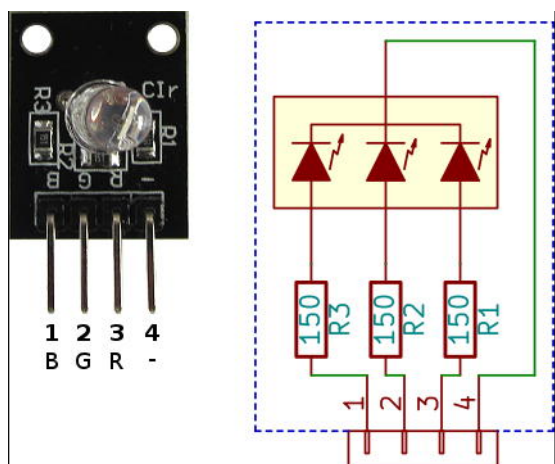


Figura 3: Módulo RGB utilizado para completar el sistema de semáforos.

El uso del módulo RGB facilita la implementación del sistema de semáforos sin la necesidad de adquirir componentes adicionales, maximizando el uso de los recursos disponibles y manteniendo el diseño del hardware lo más simple y eficiente posible.

VI. DESARROLLO Y ANÁLISIS.

En el diseño del sistema de semáforos, se ha decidido conectar los 30 LEDs en grupos de 2, donde ambos LEDs en serie representan el mismo color pero de un semáforo diferente. Esta configuración aprovecha la constante del diseño, donde los semáforos de las entradas opuestas en cada cruce siempre estarán en el mismo estado debido a la ausencia de giros a la izquierda. Al agrupar los LEDs de esta manera, se simplifica el control del sistema y se optimiza el uso de los

pins disponibles en el Arduino. Además, se utiliza un módulo RGB para completar los 2 LEDs que faltan para diseñar 16 semáforos.

Se ha agregado una pantalla LCD 20x4 y se ha configurado para mostrar los estados del sistema de descongestión. Esta pantalla permite visualizar de manera clara y detallada los estados actuales de la máquina de estados, así como las pruebas de simulación que se están ejecutando. La integración de la pantalla LCD añade una capa adicional de complejidad al proyecto, pero mejora significativamente la capacidad de monitoreo y depuración del sistema.

A continuación, se presenta una imagen del hardware del proyecto:



Figura 4: Configuración del hardware del sistema de semáforos.

El sistema de semáforos está estructurado utilizando clases y enumeraciones para gestionar los diferentes estados y comportamientos de los semáforos en los cruces. A continuación se describe brevemente la estructura y función de estas clases y enumeraciones clave.

VI-A. *Clases del sistema*

El sistema utiliza una enumeración ('enum Estado') para definir los posibles estados del semáforo: 'Cruce_H', 'Cruce_V', y 'Cambio'. Estos estados representan las fases de cruce horizontal, cruce vertical, y la señal de cambio, respectivamente.

La clase 'Semáforo' es fundamental para la gestión de los LEDs que representan los semáforos en cada cruce. Esta clase incluye las siguientes variables y métodos importantes:

- **Variables de instancia:** 'rojoNorteSur', 'verdeNorteSur', 'rojoOesteEste', y 'verdeOesteEste'. Estas variables representan los pines del Arduino que controlan los LEDs para las direcciones Norte-Sur y Oeste-Este.
- **Constructor:** Inicializa los pines de los LEDs y los configura como salidas.

- **Método ‘FSM_Semaforo’:** Este método implementa la máquina de estados finita (FSM) del semáforo. Dependiendo del estado (‘S_State’) recibido como parámetro, el método ajusta los estados de los LEDs para simular el comportamiento del semáforo en los cruces horizontales y verticales.

La función de señal de cambio (‘Cambio’) no se implementa completamente en este diseño, ya que los cruces se gestionan directamente a través de la clase ‘Semaforo’. Esta simplificación permite un control más directo y eficiente de los estados del semáforo, evitando transiciones intermedias que no son necesarias en este contexto.

VI-B. Modo normal

Una de las funciones principales del sistema es el ‘controladorIdle’, que gestiona los estados de los semáforos cuando no se detecta ninguna congestión vial. Esta función alterna entre los cruces horizontales y verticales, asegurando un flujo de tráfico equitativo en todas las direcciones.

En la primera fase, se configuran los estados de los cruces A, B, C y D para los cruces horizontales (‘Cruce_H’), activando los semáforos correspondientes durante 5 segundos. Luego, se configuran los estados de los cruces para los cruces verticales (‘Cruce_V’), activando nuevamente los semáforos correspondientes durante 5 segundos.

Este ciclo de alternancia asegura un flujo de tráfico equitativo en todas las direcciones cuando no hay congestión detectada, proporcionando tiempos de paso adecuados para cada cruce.

Para simular la cantidad de vehículos en cada carril y las conexiones entre los cruces, se utilizan varios contadores que permiten monitorear el flujo de tráfico en cada entrada y salida de las intersecciones, así como en las vías compartidas entre los cruces.

- **Entradas y salidas de cruces:** Cada cruce (A, B, C, y D) tiene contadores específicos para sus entradas y salidas. Por ejemplo, ‘Entrada_Norte_A’ y ‘Salida_Norte_A’ registran el número de vehículos que entran y salen del cruce A por el lado norte, respectivamente. De manera similar, se gestionan las entradas y salidas oeste, este, y sur para cada cruce.
- **Vías compartidas:** Las conexiones entre los cruces se gestionan mediante contadores que registran el flujo de vehículos en las calles compartidas. Por ejemplo, ‘Via_AE_BO’ representa la calle que conecta el cruce A en dirección este con el cruce B en dirección oeste, mientras que ‘Via_AS_CN’ gestiona el flujo de A sur a C norte.

Estos contadores son esenciales para simular el tráfico y tomar decisiones sobre el estado de los semáforos, ayudando a mantener un flujo eficiente y evitar congestiones en los cruces.

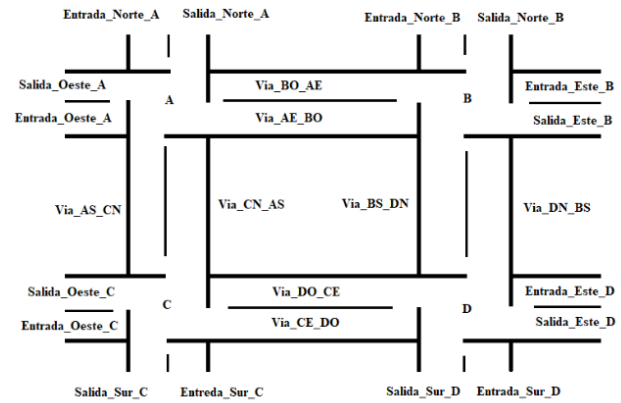


Figura 5: Diagrama de las entradas y vías entre los cruces A, B, C y D.

VI-C. Cruces_FSM()

La FSM Cruces_FSM() tiene un estado principal STATE_CONGEST que puede estar en uno de los siguientes estados:

IDLE: En este estado, se monitorea constantemente el flujo de vehículos en todas las entradas y vías compartidas entre los cruces. Si se detecta una congestión, es decir, si alguna de las contadores de vehículos supera el límite de 12 vehículos, el estado cambia a WARNING. Si no hay congestión, el sistema permanece en el modo normal (controladorIdle()) para mantener el flujo de tráfico equitativo.

WARNING: Cuando se detecta congestión en alguna de las vías, el sistema cambia al estado WARNING. En este estado, se evalúa específicamente qué cruces están experimentando congestión basándose en los contadores de vehículos. Dependiendo de la congestión detectada (por ejemplo, congestión horizontal en DESCON_H1 o DESCON_H2, y congestión vertical en DESCON_V1 o DESCON_V2), el sistema ajusta los semáforos correspondientes para mitigar el problema.

DESCON_H1, DESCON_H2, DESCON_V1, DESCON_V2: Estos estados se activan cuando se identifica congestión en vías específicas entre los cruces. Cada estado desencadena acciones específicas para controlar los semáforos y gestionar la congestión en las direcciones afectadas. Por ejemplo, DESCON_H1 y DESCON_H2 manejan congestiones horizontales, mientras que DESCON_V1 y DESCON_V2 manejan congestiones verticales.

El sistema utiliza una pantalla LCD para mostrar mensajes informativos sobre el estado actual del sistema de control de tráfico, como "WARNING", "DESCON_H1", "DESCON_H2", "DESCON_V1", o "DESCON_V2". Estos mensajes ayudan a los operadores o usuarios a entender qué acciones están tomando los semáforos en respuesta a las condiciones de tráfico.

VI-D. *set_condiciones_iniciales()*

La función `set_condiciones_iniciales()` agrega estados iniciales al sistema para inicializar la simulación al establecer los contadores que representan el flujo de vehículos en cada entrada, salida y vía compartida entre los cruces A, B, C y D. Esta inicialización es crucial para iniciar la simulación del tráfico con datos iniciales realistas, permitiendo que el sistema de control de cruces reaccione adecuadamente a los cambios dinámicos en el número de vehículos en cada punto del sistema.

1. Entradas y Salidas por cada cruce: Cada variable representa el número inicial de vehículos que ingresan (Entrada) y salen (Salida) de cada dirección en los cruces A, B, C, y D. Por ejemplo, `Entrada_Norte_A` tiene inicialmente 13 vehículos ingresando desde el norte hacia el cruce A, y `Salida_Norte_A` tiene 2 vehículos saliendo desde el cruce A hacia el norte.
2. Vías compartidas entre los cruces: Se definen variables para las vías compartidas entre los cruces, indicando el número inicial de vehículos que circulan por esas vías. Por ejemplo, `Via_AE_BO` representa la vía que va de A este a B oeste, con 5 vehículos inicialmente.

VI-E. *set_condiciones_iniciales()*

Inicializa los contadores para simular el tráfico en cada calle y las conexiones entre los cruces. Función `move_E_to_S(int &Entrada, int &Carril, int &Salida)`:

Esta función mueve un carro desde la entrada (Entrada) hacia el carril (Carril) y luego hacia la salida (Salida), decrementando los contadores correspondientes. Función `salidas_update()`:

Actualiza los contadores de salida para cada calle, decrementándolos si son mayores que cero. Función `car_move(int STATE_TO_MOVE)`:

Controla el movimiento de los carros dependiendo del estado (`STATE_TO_MOVE`) del sistema de control de cruces (Cruce_H o Cruce_V), llamando a `move_E_to_S` para cada calle de entrada y actualizando las salidas al final de cada movimiento. Este código proporciona la estructura básica y las funciones necesarias para simular el movimiento de carros entre las entradas, carriles y salidas de los cruces simulados, manteniendo un control eficiente del tráfico vehicular.

VII. CONCLUSIONES Y RECOMENDACIONES.

VII-A. Conclusiones

- Se logró implementar un sistema físico capaz de simular y controlar simultáneamente cuatro cruces, lo cual ha demostrado ser efectivo en la reducción de la congestión vial. Este logro representa un avance significativo en la gestión del tráfico urbano, mejorando la fluidez vehicular y la seguridad en los cruces.
- A pesar de los esfuerzos dedicados, no fue posible implementar la predicción por Holt-Winters debido a

limitaciones de memoria y restricciones de tiempo. Estos factores obstaculizaron la fase de implementación de este componente crucial del proyecto. La planificación inicial de sustituir esta técnica con un dashboard en Things-Board también enfrentó desafíos adicionales, incluyendo la complejidad añadida al ser desarrollado por una sola persona en lugar de un equipo, lo cual limitó aún más la viabilidad de la solución alternativa propuesta.

- El desarrollo de este laboratorio ha proporcionado una experiencia invaluable en la implementación de sistemas reales que involucran múltiples componentes interconectados. La investigación extensa y la aplicación práctica de los conocimientos adquiridos a lo largo de la carrera fueron esenciales para enfrentar los desafíos técnicos y de diseño encontrados. Este proyecto ha fortalecido nuestra capacidad para abordar problemas de ingeniería en sistemas simulados y reales, preparándonos mejor para enfrentar desafíos similares en el futuro profesional.

VII-B. Recomendaciones

- Es importante comenzar con tiempo cuando se trabaja con Hardware, los problemas pueden seguir muy rápido y los componentes deben ser revisados.
- Se debe mejorar el sistema, la implementación es básica pero muy extensa por la cantidad de dispositivos que se controlan y la cantidad de posibles estados que representan estos dispositivos, el sistema controla los estados y hace predicciones básicas basada en condiciones del sistema.
- Para mejorar la visualización de datos se debe completar el dashboard de Thinksboard.

REFERENCIAS

- [1] *LiquidCrystal I2C Library*. Retrieved from https://github.com/johnrickman/LiquidCrystal_I2C (John Rickman).