Laboratorio II

*Nota: Trabajo práctico para la unidad curricular Sistemas Embebidos

Tomás Eduardo Conti
Ing. Mecatrónica
Universidad Tecnológica del Uruguay
Fray Bentos, Río Negro, Uruguay
tomas.conti@estudiantes.utec.edu.uy

Pablo Kevin Pereira
Ing. Mecatrónica
Universidad Tecnológica del Uruguay
Fray Bentos, Río Negro, Uruguay
pablo.pereira.p@estudiantes.utec.edu.uy

***Resumen*—Se presenta el análisis temporal de un sistema de riego automatizado basado en un microcontrolador ATMega328 (Arduino UNO). Se midieron experimentalmente los retardos asociados a tareas críticas empleando instrumentación en tiempo real con señales de prueba y osciloscopio. El mayor retardo, cercano a 70 ms, corresponde a la cadena relé–electroválvula–flujo; la conmutación de banderas hardware fue prácticamente instantánea. Se propone una separación de tareas que habilita una estrategia básica de tiempo real, garantizando el determinismo del sistema.**

***Index Terms*—Sistemas embebidos, riego automático, tiempo real, ATMega328, instrumentación.**

I. Introducción

Los sistemas embebidos constituyen una base fundamental de la automatización industrial, ...

II. Materiales y Métodos

Los materiales utilizados para este laboratorio fueron componentes electrónicos de propósito general, sensores y actuadores compatibles con microcontroladores, seleccionados específicamente para evaluar el desempeño temporal de tareas críticas en un sistema de riego automatizado basado en un microcontrolador.

Para la implementación y medición experimental se hizo uso, de forma más específica, de los siguientes componentes:

- 1 × Arduino UNO (microcontrolador ATMega328)
- 1 × Osciloscopio digital SDS1000DL+ Series
- 1 × Juego de cables de prueba para osciloscopio
- 1 × Electroválvula 12V para Arduino
- 1 × Placa de prototipado (protoboard)
- 1 × Juego de cables DuPont (M–M / M–F)
- 1 × Sensor de caudal YF-S201 con efecto Hall
- 1 × Resistencia de 10 kΩ
- 1 × Pantalla LCD1602 con interfaz I2C
- 1 x Módulo de relé de 5V
- 1 x Fuente regulable de laboratorio de 24V

II-A. Metodología Experimental

El procedimiento consistió en la instrumentación de distintos puntos clave del sistema utilizando pines digitales del Arduino UNO, con el objetivo de generar señales de control que pudieran ser visualizadas y medidas mediante osciloscopio.

Este trabajo se desarrolló en el laboratorio del ITR Suroeste, UTEC.

Las mediciones se realizaron de la siguiente manera:

- Se utilizó una señal digital de prueba para evaluar el tiempo de activación de un pin digital a un estado alto.
- Se midió el tiempo de escritura de mensajes en la pantalla LCD a través de cambios en un pin configurado como bandera durante la función de impresión.
- Para determinar el retardo entre la activación de la válvula y el inicio del flujo de agua, se usó el sensor de caudal en línea con la válvula y se comparó la señal de activación del relé con la primera aparición de pulsos del sensor YF-S201.
- La duración total del ciclo de riego fue medida envolviendo el inicio y el fin del ciclo en una señal digital.
- Se evaluó también la duración de las rutinas de decisión en diferentes escenarios (suelo seco vs húmedo), midiendo el tiempo entre la lectura de los sensores y la ejecución de la decisión lógica.

Las señales fueron capturadas con el osciloscopio digital, garantizando una observación precisa de los eventos. El análisis permitió caracterizar el comportamiento temporal del sistema y proponer un esquema de separación de tareas adecuado para operación en tiempo real.

II-B. Módulo 1: Medición de banderas de Hardware

Con el objetivo de conocer el tiempo que demora el microcontrolador en ejecutar una secuencia de instrucciones simples y reflejar los cambios en los pines digitales, se implementó una rutina de prueba que manipula el estado de un pin de salida del Arduino UNO.

Se utilizó el pin digital 2 del microcontrolador como señal de medición (`Measure_Pin`), y mediante el siguiente fragmento de código se generó una transición rápida entre niveles lógicos:

```
// ==== Lectura de cambios de estado ====
void measure_1(){
  digitalWrite(Measure_Pin, HIGH);
  digitalWrite(Measure_Pin, LOW);
}
```

Este cambio rápido produce una señal cuadrada de corta duración, cuya observación mediante osciloscopio permite visualizar el retardo entre instrucciones de escritura digital y su efecto en el pin físico.

II-C. Módulo 2: Delay en la lectura del sensor de humedad

Con el objetivo de cuantificar el tiempo requerido por el sistema para realizar la lectura del sensor capacitivo de humedad del suelo, se implementaron dos aproximaciones experimentales utilizando la técnica de marcación mediante un pin digital y visualización en osciloscopio.

II-C1. Primera medición: lectura directa con `analogRead()`: En esta primera prueba se midió exclusivamente el tiempo que demora el microcontrolador en ejecutar una lectura directa del ADC mediante la instrucción `analogRead(MoisturePin)`. Para ello, se instrumentó el pin de medición (`Measure_Pin`) envolviendo el acceso ADC entre señales de activación y desactivación:

```
// ==== Lectura de Sensor directa ====
void measure_2(){
  digitalWrite(Measure_Pin, HIGH);
  analogRead(MoisturePin);
  digitalWrite(Measure_Pin, LOW);
}
```

II-C2. Segunda medición: lectura con procesamiento adicional: En esta segunda prueba, se agregó una función de lectura que, además de obtener el valor crudo del ADC, realiza un procesamiento para convertirlo a una escala porcentual. Se buscó determinar si la inclusión de una operación matemática adicional afectaba significativamente el tiempo de ejecución.

```
// ==== Lectura de Sensor procesada ====
long readMoisture() {
  long value = (analogRead(MoisturePin)/1023)*100;
  return value;
}
```

II-D. Módulo 3: Tiempo de ejecución de escritura de mensajes en la LCD

Con el objetivo de determinar el impacto temporal asociado a la actualización del contenido de la pantalla LCD1602, se realizaron dos aproximaciones experimentales utilizando técnicas de marcación mediante un pin digital, visualización con osciloscopio y evaluación de frecuencia de refresco.

II-D1. Primera medición: actualización completa de la LCD: En esta prueba se evaluó el tiempo requerido para borrar completamente la pantalla y reescribir todo su contenido en ambas líneas. La rutina `updateAll_LCD()` realiza un `lcd.clear()` seguido de múltiples líneas de impresión con formateo.

```
void measure_3(){
  digitalWrite(Measure_Pin, HIGH);
  updateAll_LCD();
  digitalWrite(Measure_Pin, LOW);
  delay(30);
}

void updateAll_LCD() {
  unsigned long start = micros();
  lcd.clear();
  char raw1[16];
  snprintf(raw1, sizeof(raw1),
       "M:%c,P:%is,HL:%is",
       Mode, IrrigationPeriod, IrrigationTime);
  lcd.print(raw1);
  lcd.setCursor(0, 1);
  char raw2[16];
  if (currMoisture>99){ currMoisture = 99; }
  snprintf(raw2, sizeof(raw2),
        "mMois:%02ld%%,C:%02ld%%",
        MoistureThreshold, currMoisture);
  lcd.print(raw2);
  unsigned long finish = micros();
  Serial.print("3. Tiempo de ejecucion de escritura
      de mensajes en la LCD");
  Serial.print(finish - start);
}
```

II-D2. Segunda medición: actualización optimizada de campos individuales: En esta segunda prueba se utilizó una estrategia optimizada en la cual solo se actualizan los campos modificados del display. La función 'updateLCD()' posiciona el cursor en ubicaciones específicas y sobrescribe los valores, evitando borrar y reescribir el contenido completo.

```
void measure_3b(){
  digitalWrite(Measure_Pin, HIGH);
  updateLCD();
  digitalWrite(Measure_Pin, LOW);
  delay(5);
}

void updateLCD() {
  char buf[2];
  lcd.setCursor(2, 0);
  snprintf(buf, sizeof(buf), "%c", Mode);
  lcd.print(buf);
  lcd.setCursor(6, 0);
  snprintf(buf, sizeof(buf), "%i",
      IrrigationPeriod);
  lcd.print(buf);
  lcd.setCursor(13, 0);
  snprintf(buf, sizeof(buf), "%i", IrrigationTime);
  lcd.print(buf);

  lcd.setCursor(6, 1);
  snprintf(buf, sizeof(buf), "%.02f",
      MoistureThreshold*100/1023);
  lcd.print(buf);
  lcd.setCursor(12, 1);
  snprintf(buf, sizeof(buf), "%02ld",
      currMoisture*100/1023);
  lcd.print(buf);
}
```

II-E. Módulo 4: Tiempo de apertura efectiva de la válvula

El objetivo de esta medición fue estimar el tiempo real que transcurre entre la activación de la válvula de riego y el inicio efectivo del paso de agua. Esta latencia incluye tanto el retardo de actuación del relé como el tiempo de apertura mecánica de la electroválvula.

Se utilizó un módulo de relé de 5V controlado por el pin digital 2 del Arduino UNO. La electroválvula de 12V fue conectada al contacto normalmente abierto del relé, y alimentada mediante una fuente externa regulada a 12V. En serie con la válvula se instaló un caudalímetro YF-S201, que entrega una señal de pulsos proporcional al caudal instantáneo.

El osciloscopio digital se configuró en modo de doble canal, conectando el canal CH1 a la salida del pin de control del

relé (trigger) y el canal CH2 a la salida de pulsos del sensor de caudal. La señal de trigger representa el instante en que la válvula comienza su apertura, mientras que los pulsos del sensor indican el inicio del paso de agua.

La rutina utilizada para generar la señal de prueba se muestra a continuación:

```
// ==== Activacion de la valvula para prueba ====
void measure_4(){
digitalWrite(Measure_Pin, HIGH);
delay(1000); // Tiempo con valvula abierta
digitalWrite(Measure_Pin, LOW);
delay(3000); // Tiempo con valvula cerrada
}
```

II-F. Módulo 5: Tiempo de ejecución del ciclo completo

En esta etapa del análisis se evaluó el tiempo de ejecución del ciclo completo de riego en condiciones de vacío, es decir, sin interacción efectiva con el hardware físico como la válvula o el flujo de agua. El objetivo fue determinar el tiempo de procesamiento del programa por sí solo, aislando posibles cuellos de botella internos y evaluando su impacto sobre la periodicidad de actualización.

Se utilizó nuevamente el pin de medición conectado al osciloscopio para marcar la duración del ciclo de ejecución, comprendido desde la adquisición del dato hasta la escritura de los resultados en la pantalla LCD. La rutina se midió utilizando el siguiente código:

```
//Medicion del ciclo completo
void loop() {
//digitalWrite(Measure_Pin, HIGH);

readSwitches();
unsigned long now = millis();
if (now - last_time > samplePeriod){
currMoisture = readMoisture();
controlIrrigation(currMoisture);
updateAll_LCD();
unsigned long finish = millis();
Serial.print("5. Tiempo de ejecucion del ciclo
    completo (ms): ");
Serial.println(finish - now);
}
}
```

II-G. Módulo 6: Evaluación de la rutina de decisión

Si bien se consideró inicialmente evaluar el impacto temporal de la rutina de decisión del sistema de riego —es decir, el conjunto de comparaciones lógicas y bifurcaciones que determinan si se debe o no activar la válvula según la humedad leída—, las pruebas experimentales mostraron que dicho bloque no genera variaciones significativas en los tiempos de ejecución. Esto se comprobó indirectamente al comparar las duraciones del ciclo completo con y sin ejecución efectiva del riego, observándose una latencia prácticamente idéntica entre ambos casos. Dado que las operaciones lógicas involucradas consisten en comparaciones simples y asignaciones condicionales (por ejemplo, `if` anidados), se concluyó que su impacto temporal es despreciable frente al resto de los procesos. Por esta razón, y al no haberse registrado diferencia observable mediante osciloscopio, se optó por no realizar una medición dedicada a esta rutina.

III. Resultados

III-A. Módulo 1: Medición de banderas de Hardware

En la Fig. 1 se presenta la forma de onda obtenida en el canal CH1 del osciloscopio digital. La señal alterna entre niveles alto y bajo, con un periodo total de 8,51 μs. Se registraron los siguientes valores relevantes:

- Tiempo en nivel alto (+Wid): 4,06 μs
- Tiempo en nivel bajo (-Wid): 4,45 μs
- Tiempo de subida (rise time): 30,00 ns
- Tiempo de bajada (fall time): 40,00 ns
- Frecuencia: 116,756 kHz

El valor de +Wid representa el tiempo que la señal permaneció en nivel alto, es decir, el tiempo que tomó la ejecución de la instrucción `digitalWrite(Measure_Pin, LOW);` tras haber sido activada la salida. De forma análoga, el tiempo en nivel bajo representa la ejecución de la instrucción que restablece el pin a nivel alto.

Estos resultados permiten estimar que la latencia entre ambas instrucciones es de aproximadamente 4 μs, lo cual representa una demora mínima y aceptable para tareas de control digital en sistemas embebidos no críticos.

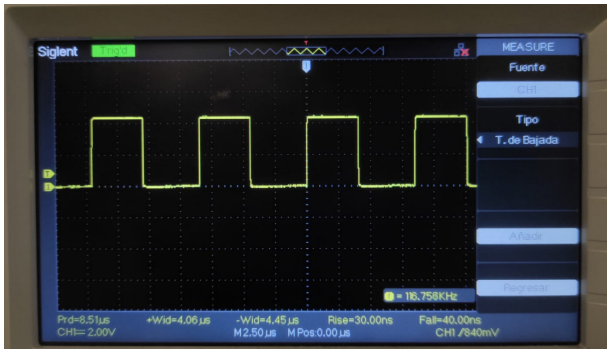

Figura 1. Captura del osciloscopio mostrando la medición de tiempo de ejecución entre dos instrucciones consecutivas de cambio de estado digital.

III-B. Módulo 2: Medición de valores analógicos

III-B1. Medición directa analógica: La forma de onda obtenida se muestra en la Fig. 2. Se registraron los siguientes parámetros:

- Periodo total: 120,0 μs
- Tiempo en alto (+Wid): 115,8 μs
- Tiempo en bajo (-Wid): 4,44 μs
- Tiempo de subida/bajada: 80 ns

III-B2. Medición procesada analógica: La forma de onda obtenida se muestra en la Fig. 3. Se registraron los siguientes parámetros:

- Periodo total: 120,2 μs
- Tiempo en alto (+Wid): 115,7 μs
- Tiempo en bajo (-Wid): 4,44 μs

Figura 2. Captura del osciloscopio para lectura directa con `analogRead()`.

- Tiempo de subida/bajada: 80 ns

Figura 3. Captura del osciloscopio para lectura con función `readMoisture()`.

III-B3. Análisis comparativo: Ambas mediciones arrojaron resultados prácticamente idénticos, lo que sugiere que el procesamiento adicional —en este caso una división y multiplicación entera— no genera una carga computacional significativa para el microcontrolador ATMega328. La lectura del sensor de humedad puede estimarse con una latencia inferior a los 116 μs, incluyendo tanto el acceso ADC como el procesamiento numérico posterior.

Este dato es fundamental para el diseño de una arquitectura en tiempo real, ya que permite calendarizar con seguridad la tarea de adquisición de humedad dentro de ventanas temporales estrechas.

III-C. Módulo 3: Tiempo de ejecución de escritura de mensajes en la LCD

III-C1. Primera medición: actualización completa de la LCD: Los resultados observados se muestran en la Fig. 4. Se midieron los siguientes parámetros:

- Periodo total: 70,82 ms
- Tiempo en nivel alto: 40,80 ms
- Tiempo en nivel bajo: 30,04 ms
- Tiempo de subida/bajada: 40 μs

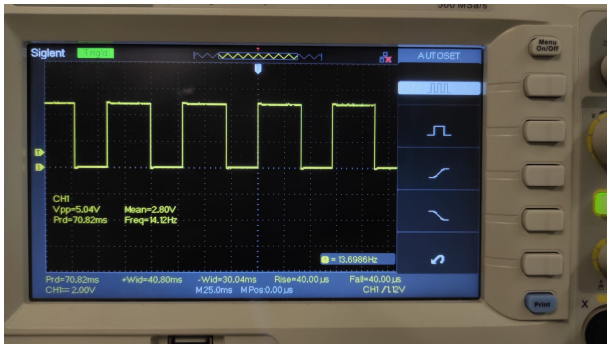

Figura 4. Medición del tiempo de escritura total en la pantalla LCD1602.

III-C2. Segunda medición: actualización optimizada de campos de texto individuales: La señal observada en el osciloscopio para esta mejora se muestra en la Fig. 5. Los parámetros medidos fueron:

- Periodo total: 8,12 ms
- Tiempo en nivel alto: 3,11 ms
- Tiempo en nivel bajo: 5,01 ms
- Tiempo de subida/bajada: 4 μs

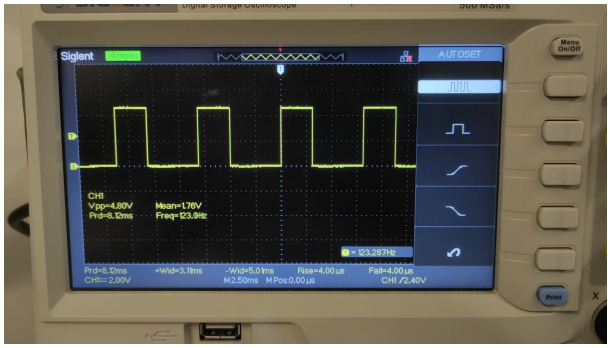

Figura 5. Medición con actualización optimizada de campos individuales en la LCD.

III-C3. Análisis comparativo: La diferencia entre ambas estrategias de actualización de pantalla es significativa. Mientras que la actualización completa requiere más de 40 ms, el método optimizado reduce el tiempo a solo 3 ms, evidenciando un ahorro superior al 90 %. Esta mejora puede tener un impacto positivo en sistemas donde la LCD comparte recursos o interrupciones con otras tareas críticas, contribuyendo a la eficiencia general del sistema embebido.

III-D. Módulo 4: Tiempo de apertura efectiva de la válvula

La forma de onda obtenida muestra una separación temporal clara entre la activación del relé (borde ascendente en CH1) y la aparición del primer pulso en la salida del caudalímetro (CH2). El retardo medido visualmente en el osciloscopio fue de aproximadamente 70,ms, valor que contempla el tiempo de respuesta de los componentes mecánicos involucrados.

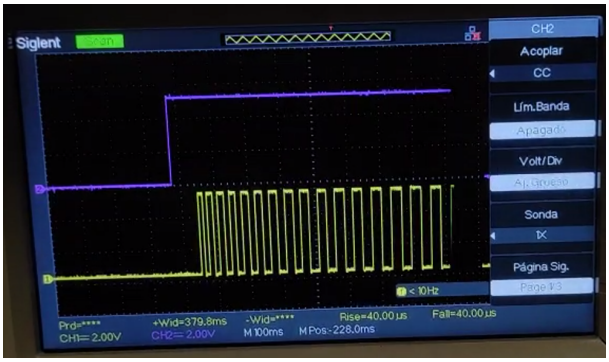

Figura 6. Medición del tiempo entre activación del relé (CH1) y respuesta del caudalímetro (CH2).

Este retardo representa el mayor tiempo de latencia observado en todo el sistema y debe ser considerado al momento de planificar tareas críticas de control. No obstante, su carácter determinista permite compensarlo mediante técnicas simples de programación en arquitectura de tiempo real.

III-E. Módulo 5: Tiempo de ejecución del ciclo completo

La forma de onda obtenida en el osciloscopio se muestra en la Fig. 7. Se identificaron los siguientes parámetros relevantes:

- Periodo total: 8,48,ms
- Tiempo en alto (+Wid): 3,46,ms
- Tiempo en bajo (-Wid): 5,02,ms
- Tiempo de subida/bajada: 4,μs

Figura 7. Captura del osciloscopio durante la ejecución del ciclo completo del programa, en vacío.

III-E1. Análisis del ciclo: El tiempo total de ejecución programática del ciclo de riego, sin considerar el tiempo de apertura de la válvula, es de aproximadamente 8,5,ms. Este valor refleja únicamente el procesamiento de datos, lectura de sensores, lógica de control y actualización de pantalla.

En aplicaciones reales, a este tiempo debe sumarse el retardo físico de apertura de la válvula, estimado previamente en 70,ms, llevando el tiempo total del ciclo a aproximadamente 78,ms.

Este análisis permite identificar que el mayor cuello de botella no se encuentra en el software sino en la respuesta mecánica de los actuadores. Aun así, el tiempo de ejecución lógico del ciclo sigue siendo un factor clave para la planificación temporal en sistemas embebidos con múltiples tareas concurrentes.

Discusión global de restricciones y propuesta de planificación

El análisis temporal realizado permite identificar con claridad las limitaciones extremas del sistema. **(i) Restricción más corta.** La operación que impone la exigencia temporal más estricta es la conmutación de una bandera de `digitalWrite()`, que presentó una latencia total de ~4,µs entre flancos. Esta cifra fija un umbral mínimo: cualquier tarea que requiera reaccionar a eventos externos deberá ejecutarse en, al menos, ese orden de magnitud para no perder muestras ni generar metastabilidad.

(ii) Restricción más larga. En el extremo opuesto, la apertura efectiva de la electroválvula —incluyendo retardo del relé y tiempo mecánico— arrojó un retardo de ~70,ms antes de que el caudalímetro registre flujo. Esta es, por lejos, la restricción temporal dominante y se convierte en el factor crítico al calendarizar el ciclo de riego.

(iii) Tarea más importante. A la luz de lo anterior, la tarea con mayor impacto funcional es el *control de riego*: decidir, activar y supervisar la apertura/cierre de la válvula en base a la humedad del suelo. Un error o demora excesiva aquí repercute directamente en el consumo de agua y en la salud de la planta, mientras que retrasos en la interfaz de usuario (LCD) o en el registro de datos sólo afectan la experiencia, pero no la misión principal.

(iv) Propuesta de separación de tareas para una ejecución en tiempo real básica.

- *Tarea de alta frecuencia* (~1 kHz, prioridad 1): lectura del sensor de humedad y detección de pulsos del caudalímetro. Su ventana de ejecución es de <120,µs, muy por debajo del periodo, por lo que cabe holgura suficiente.
- *Tarea de control* (~20 Hz, prioridad 2): algoritmo de decisión y disparo de la válvula. Al asignarle una frecuencia cinco veces menor que la anterior se asegura que siempre disponga de una muestra fresca antes de actuar y que los 70,ms de latencia mecánica queden absorbidos dentro de su propio ciclo.
- *Tarea de interfaz* (~2 Hz, prioridad 3): actualización parcial de la LCD (método optimizado de 3 ms). Con esta cadencia no interfiere con las acciones de control y mantiene la pantalla fluida para el usuario.
- *Tarea de baja prioridad* (en *idle*): registro en memoria/-serial y procesamiento estadístico. Se ejecuta únicamente cuando el procesador queda ocioso.

Esta división permite emplear un *scheduler* cooperativo sencillo (o bien la combinación de temporizadores e interrupciones nativas del ATMega328), garantizando que:

1. todas las tareas de tiempo crítico finalicen antes de su siguiente activación,
2. la decisión de riego siempre disponga de datos actualizados,
3. la latencia larga y determinista de la válvula quede contenida dentro de un margen temporal conocido, ha-

bilitando así una operación *soft real-time* suficiente para la aplicación propuesta.

El siguiente enlace lleva a una carpeta de Drive con los archivos multimedia y de código propios del proyecto.
https://drive.google.com/drive/folders/1nDGLTFtttrkeJWOV4I1NXmuiVmMxCGDn?usp=drive_link

REFERENCIAS

[1] Quiroga, D. (2025). *Sistemas embebidos (Séptimo semestre IMEC, 2025-1S)*. UTEC – Instituto Tecnológico Regional Suroeste.