Universidad Nacional de San Martín

Trabajo Práctico Final

Laboratorio de computación 1

Docentes:

Pedro Facundo Iriso

Matias Jose Gagliardo

GRUPO 11.

Lucas Viña

Julian Rivadeo

Juan Ignacio Cucarella

Maximiliano Candia

segundo cuatrimestre 2025

• Resumen del proyecto

Para este proyecto se simulará un paso peatonal para ciegos, adaptado a una avenida con 3 cruces peatonales, y una alarma sonora que se emitirá a petición del peatón cuando decida cruzar la calle, de esta manera la persona no vidente usará debidamente el paso mediante el sentido de la audición.

Los semáforos serán realistas y continuos. Los botones de cada cruce colocarán el semáforo a rojo durante determinado tiempo.

• Descripción técnica del hardware

1 - Placa Arduino UNO

El Arduino Uno es una placa de microcontrolador de dispositivos basado en el microchip ATmega328 y desarrollado por Arduino. La placa está equipada con conjuntos de pines de E/S digitales y analógicas que pueden conectarse a varias placas de expansión y otros circuitos. La placa tiene 13 pines digitales, 6 pines analógicos y programables con el Arduino IDE (Entorno de desarrollo integrado) a través de un cable USB tipo B.Puede ser alimentado por el cable USB o por una batería externa de 9 voltios, aunque acepta voltajes entre 7 y 20 voltios.

Ficha técnica:

- Microcontrolador: Microchip ATmega328P
- Voltaje de funcionamiento: 5 voltios
- Voltaje de entrada: 7 a 12 voltios
- Pines de E/S digitales: 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica: 6
- Corriente DC por Pin de E/S: 20 mA
- Corriente CC para Pin de 3.3V: 50 mA
- Memoria Flash: 32 KB de los cuales 0.5 KB utilizados por el gestor de arranque
- SRAM: 2 KB
- EEPROM: 1 KB
- Velocidad del reloj: 16 MHz
- Longitud: 68.6mm

- Ancho: 53,4mm
- Peso: 25g

1 - Protoboard

Una placa de pruebas está compuesta por varios bloques de plástico perforados y numerosas láminas delgadas, de una aleación de cobre, estaño y fósforo, que unen dichas perforaciones, creando una serie de líneas de conducción paralelas. Las líneas se cortan en la parte central del bloque para garantizar que dispositivos en circuitos integrados de tipo dual in-line package (DIP) puedan ser insertados perpendicularmente y sin ser tocados por el proveedor a las líneas de conductores. En la cara opuesta se coloca un forro con pegamento, que sirve para sellar y mantener en su lugar las tiras metálicas.

Debido a las características de capacitancia (de 2 a 30 pF por punto de contacto) y resistencia que suelen tener las placas de pruebas están confinados a trabajar a relativamente baja frecuencia (inferior a 10 o 20 MHz, dependiendo del tipo y calidad de los componentes electrónicos utilizados).

Los demás componentes electrónicos pueden ser montados sobre perforaciones adyacentes que no compartan la tira o línea conductora e interconectados a otros dispositivos usando cables, usualmente unifilares. Uniendo dos o más placas es posible ensamblar complejos prototipos electrónicos que cuenten con decenas o cientos de componentes.

1 - Buzzer TMB1205A

El TMB1205A es un buzzer piezoeléctrico activo de 5V con un diámetro de 12mm. Este componente genera un sonido (generalmente una frecuencia de \(2.4\) kHz) cuando se le aplica una corriente de alrededor de \(28\) mA. Es del tipo "Active Buzzer" (se activa con una señal de voltaje de DC) y se instala sobre una placa de circuito impreso (PCB) usando la técnica "through-hole".

Ficha técnica: Voltaje: \(5\) V de corriente continua (DC).Corriente: Aproximadamente \(28\) mA.Frecuencia: \(2.4\) kHz.Tipo: Buzzer activo, lo que significa que genera sonido internamente al recibir el voltaje.Montaje: A través de orificios ("through-hole") en una PCB.Diámetro: \(12\) mm.

3 - Pulsadores con 4 terminales (1a,1b,2a,2b)

Un pulsador o interruptor, es un dispositivo simple con dos posiciones, ON y OFF, un ejemplo es el interruptor de la luz.

Estos pequeños pulsadores son de un 1/4 por cada lado, tienen 4 pastillas por lo que se puede pensar que hay 4 cables, pero son dos de cada lado unidos, por tanto, este pulsador es solamente de 2 cables.

Ficha técnica:

- Utilizado como switch o interruptor al momento de ser presionado.
- Funciona como contacto normalmente abierto (NA).
- Infinito número de aplicaciones.
- Aguanta hasta 50 A.
- Voltaje: 120 VDC/ 220 VAC.
- Tamaño muy reducido.
- 4 pines amigables para usar en el protoboard.

3 – Led RGB Cátodo común

Un LED RGB es una fuente de luz compuesta por tres LEDs individuales: rojo (R), verde (G) y azul (B). Cada uno de estos LEDs puede ser controlado de forma independiente para variar la intensidad de la luz que emite, lo que permite generar una amplia variedad de colores mediante la combinación de estos colores primarios en diferentes proporciones.

- Corriente típica: 20mA
- Ángulo de visión:45°
- Diámetro de encapsulado: 5mm
- Material de Shell: Epóxico
- Número de pines: 4

  1: Led Rojo (Vcc)

  2: Cátodo común (Gnd)

  3: Led Verde (Vcc)

  4: Led Azul (Vcc)

Resistencias

Barra cerámica de carbón con una resistencia eléctrica de 220 Ω, con una tolerancia del 5% y una capacidad de disipación de ¼ de Vatio.

Ficha técnica:

- Resistencia Eléctrica: 220 Ω
- Potencia de disipación: 0,25 vatios.
- Tecnología de inserción (through hole).
- Fabricante: Genérico.
- Disposición: Tipo Axial.
- Tolerancia: 5%.

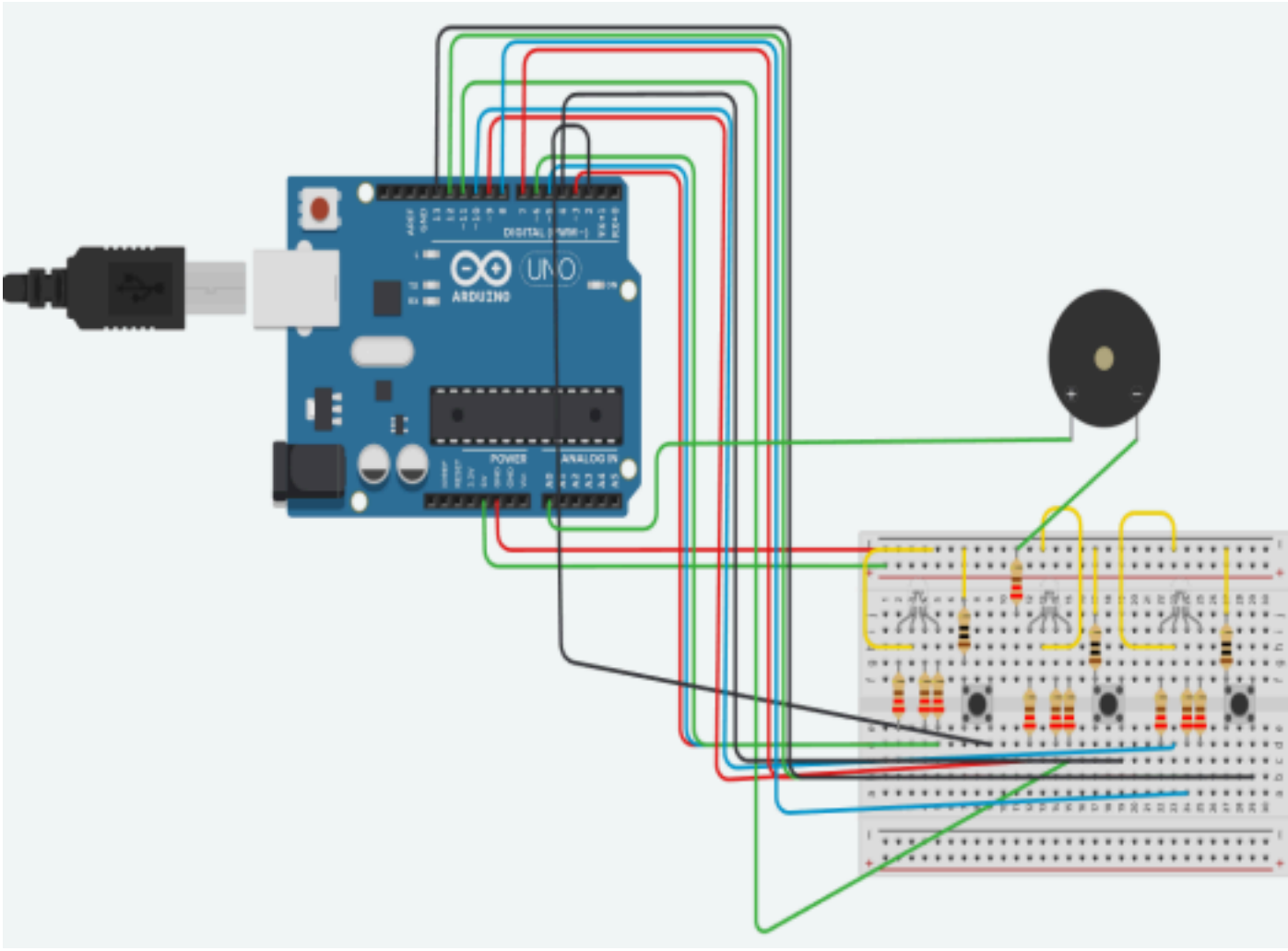
Cables macho-macho

La función del cable macho-macho es con frecuencia **usado en el tablero protoboard haciendo posible la conexión de dos elementos ingresados en dicho tablero**. Se conoce como macho-macho debido al fragmento que sobresale de los extremos del cable

Ficha técnica:

- **Tipo de Conector:** Macho - Macho (M-M).
- **Paso del Conector:** 2.54 mm (0.1 pulgadas), el estándar para protoboards.
- **Compatibilidad:** Protoboards, headers de pines (como los de Arduino).
- **Calibre del Conductor:** 22 AWG (American Wire Gauge) es el más común.
- **Material del Conductor:** Cobre (usualmente estañado).
- **Material del Aislante:** PVC o Silicona.
- **Tensión Nominal:** 30V - 50V (diseñados para lógica de bajo voltaje).
- **Corriente Nominal:** 1A - 3A (depende de la calidad del cable).
- **Rango de Temperatura:** -20°C a 85°C.
- **Longitudes Comunes:** 10 cm, 15 cm, 20 cm, 30 cm.
- **Colores:** Vienen en colores surtidos para facilitar la identificación de las conexiones (ej. rojo para VCC, negro para GND).
- **Tipos de Conductor Interno:** Pueden ser de **Núcleo Sólido** (rígido, ideal para conexiones fijas en la protoboard) o **Núcleo Flexible** (más duraderos, ideales para conectar con placas externas).

• Diagrama de conexiones



• Descripcion del software y su estructura

Definiciones (#define): Se utilizan para asignar nombres legibles a los pines físicos del Arduino (ej. R1, G1, button, BUZZER_LED).

Constantes de Tiempo: Se definen los tiempos de la onda verde (TIEMPO_VERDE, TIEMPO_ROJO, TIEMPO_AMARILLO, DESFASE)

Variables de Estado: Son las variables globales que guardan la "memoria" del sistema.

- tiempoInicio: guarda el millis() del inicio de cada ciclo de “onda verde”.
- buzzerActivo: Variable booleana (bool) que actúa como el controlador de la máquina de estados (Estado NORMAL o PEATON).
- buzzerInicio: Almacena el millis() del inicio del ciclo de peatón.
- rojoForzado1, rojoForzado2, rojoForzado3: interruptor (bool) que indican si un semáforo debe ser forzado a rojo, anulando la onda verde.
- conteoBuzzer: variable

Funciones para colores:

- colorRojo(), colorVerde(), colorAmarillo(): para encender un LED RGB en un color específico.

Función de onda verde:

- actualizarSemaforo() es reutilizable y se llama una vez por cada semáforo.

- Recibe 5 parámetros: los pines (R, G, B), su desfase de tiempo, y el interruptor forzado.

- La lógica primero comprueba el parámetro forzado. Si es true, pone el semáforo en rojo y termina (return). Esto permite que la interrupción del peatón anule la onda verde.

- Si forzado es false, la función calcula el tiempo transcurrido y usa el desfase para determinar si es el turno del semáforo de estar en verde, amarillo o rojo, implementando así la "onda verde".

Funciones setup() y loop():

- setup(): Función estándar que se ejecuta una vez. Configura todos los pines (entradas INPUT_PULLUP para los botones y OUTPUT para los LEDs y buzzer) e inicializa el Monitor Serial Serial.begin(9600).

- loop(): Es el cerebro que se ejecuta continuamente. Sus responsabilidades son:

- Leer Entradas: Obtiene el estado de los 3 botones.

- Gestionar la Máquina de Estados: Contiene los dos bloques if principales que definen el sistema:

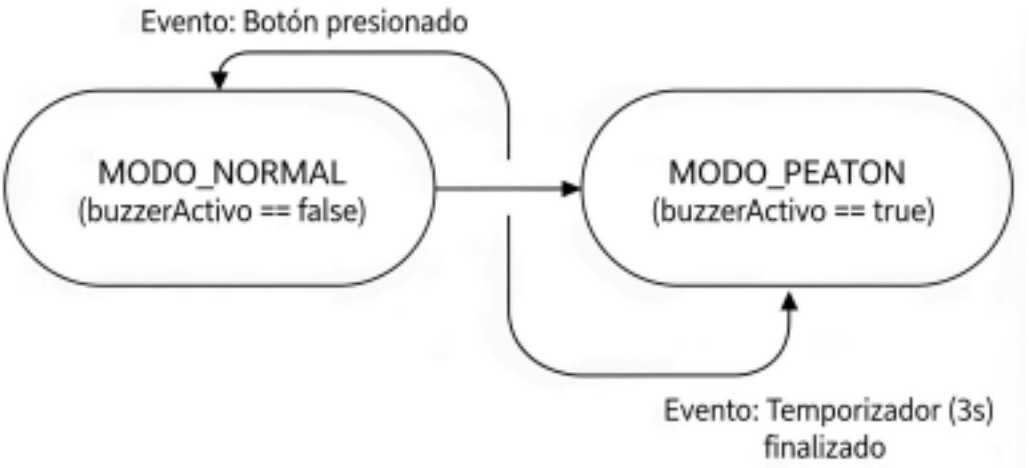
- Transición (Normal → Peatón): El if (...) && !buzzerActivo) detecta el flanco del botón, activa el buzzerActivo, guarda el buzzerInicio, activa el tone() y sube el interruptor rojoForzadoX correspondiente.

- Transición (Peatón → Normal): El if (buzzerActivo && ...) detecta el fin del temporizador de 3 segundos, apaga el buzzerActivo, el noTone() y baja los interruptores rojoForzado correspondiente.

- Gestionar Tiempos: Revisa si el cicloTotal de la onda verde ha terminado para reiniciar el tiempoInicio.

- Actualizar Salidas: Llama a actualizarSemaforo() tres veces, pasando los pines y los interruptores de estado (rojoForzado1, rojoForzado2, rojoForzado3) correspondientes a cada semáforo.

• Diagrama de máquina de estados.



• Descripcion de contador de flancos y control por tiempo.

- Variable del Contador: Se usa una variable global, conteoBuzzer = 0 para guardar el número de veces que se activa un pulsador para cortar un semaforo

- Detecta el cambio de estado del sistema de MODO_NORMAL a MODO_PEATON. Esto se logra en el loop() principal con la siguiente lógica:

```
if ((boton1_presionado || boton2_presionado || boton3_presionado)
&& !buzzerActivo) {

 buzzerActivo = true;
 buzzerInicio = ahora;
 tone(BUZZER_LED, 500);

 // Forzamos a rojo SÓLO el semáforo del botón presionado
if (boton1_presionado)
 rojoForzado1 = true;
 if (boton2_presionado)
 rojoForzado2 = true;
 if (boton3_presionado)
 rojoForzado3 = true;

 // Se cuenta e inicializa el flanco de Buzzer
 conteoBuzzer++;
 Serial.print("Activacion de buzzer numero: ");
 Serial.println(conteoBuzzer);
 }
```

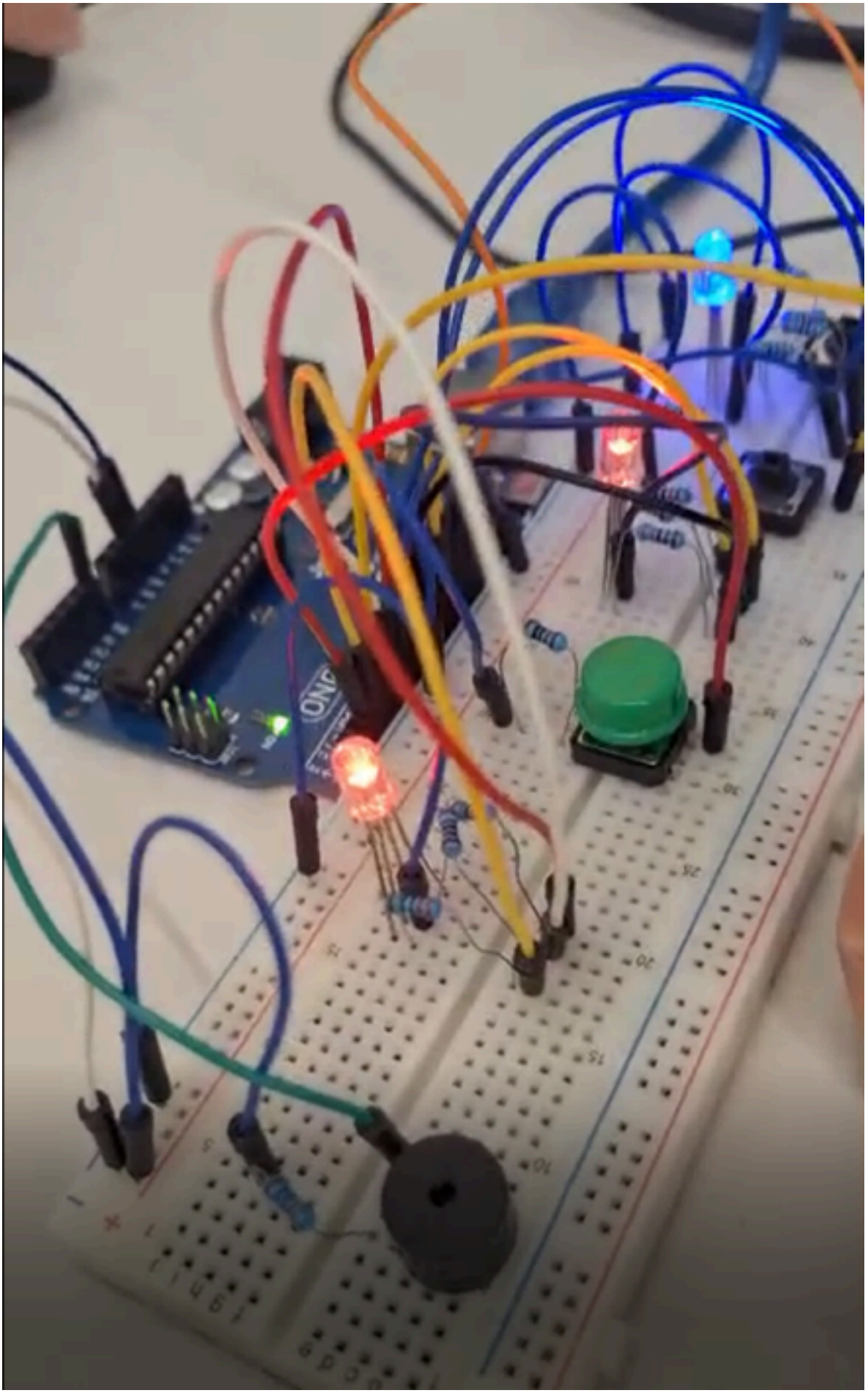
Control de tiempo de la onda verde:

- Variables: unsigned long tiempoInicio y unsigned long cicloTotal.
- Lógica: tiempoInicio es como un "cronómetro" que marca el inicio del ciclo completo de la onda verde. La función actualizarSemaforo() usa este valor (unsigned long tiempo = (millis() - tiempoInicio)) para calcular cuánto tiempo ha pasado.
- Funcionamiento: La función compara el tiempo transcurrido con el DESFASE de cada semáforo y los TIEMPO_VERDE, TIEMPO_AMARILLO, etc., para decidir qué color mostrar.
- Reinicio: El ciclo se reinicia automáticamente cuando el tiempo transcurrido alcanza el cicloTotal, actualizando tiempoInicio al valor actual de millis() sin detener el programa.

Cruce peatonal:

- Variables: unsigned long buzzerInicio y la constante de 3000ms.
- Lógica: Cuando se presiona un botón, buzzerInicio captura el valor actual de millis() en ese instante.
- Funcionamiento: El código usa esta marca de tiempo para contar 3 segundos. La condición if (buzzerActivo && (ahora - buzzerInicio >= 3000)) comprueba continuamente si el tiempo del peatón ha expirado.
- Finalización: Al cumplirse los 3 segundos, el sistema apaga el buzzer (noTone()) y desactiva las banderas (rojoForzado), volviendo al modo normal.

• Captura del sistema



● Conclusiones:

Supimos comprender la versatilidad del Arduino y a seleccionar un proyecto de acuerdo a nuestros intereses, el uso sobre distintos “cronometros” en un mismo sistema en conjunto con funciones globales con variables locales y que se relacionen entre sí para crear un semáforo para personas no videntes. Fue un desafío mezclar la electrónica y el código en algo físico y que funcione segun nuestras expectativas, este proyecto y otros dan muestra de la potencia y la importancia del lenguaje C/C++ y el software libre en los sistemas informáticos.