Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Cómputo Integrado

Practica #1: Secuenciador LEDS

Catedrático: Carlos Adrian Perez Cortez

Equipo #12

Nombre	Matricula	Carrera
Jorge Alberto Morales Reyes	1895340	ITS
Kevin Isaias Martinez Saucedo	1886121	ITS
Nelson Andres Mendez Martinez	1930671	ITS

Grupo: 003

Hora: N4

Ciudad Universitaria, San Nicolás de los Garza, Nuevo León

Fecha: 20/8/2024

Objetivo

El objetivo de esta práctica es desarrollar un secuenciador de LEDs que permite visualizar una secuencia de encendido y apagado en un patrón de "rebote" con variación dinámica del retardo de tiempo. Se implementarán dos versiones del código: una sin el uso de arreglos ni funciones, y otra utilizando arreglos y funciones.

Hallazgos y Dificultades

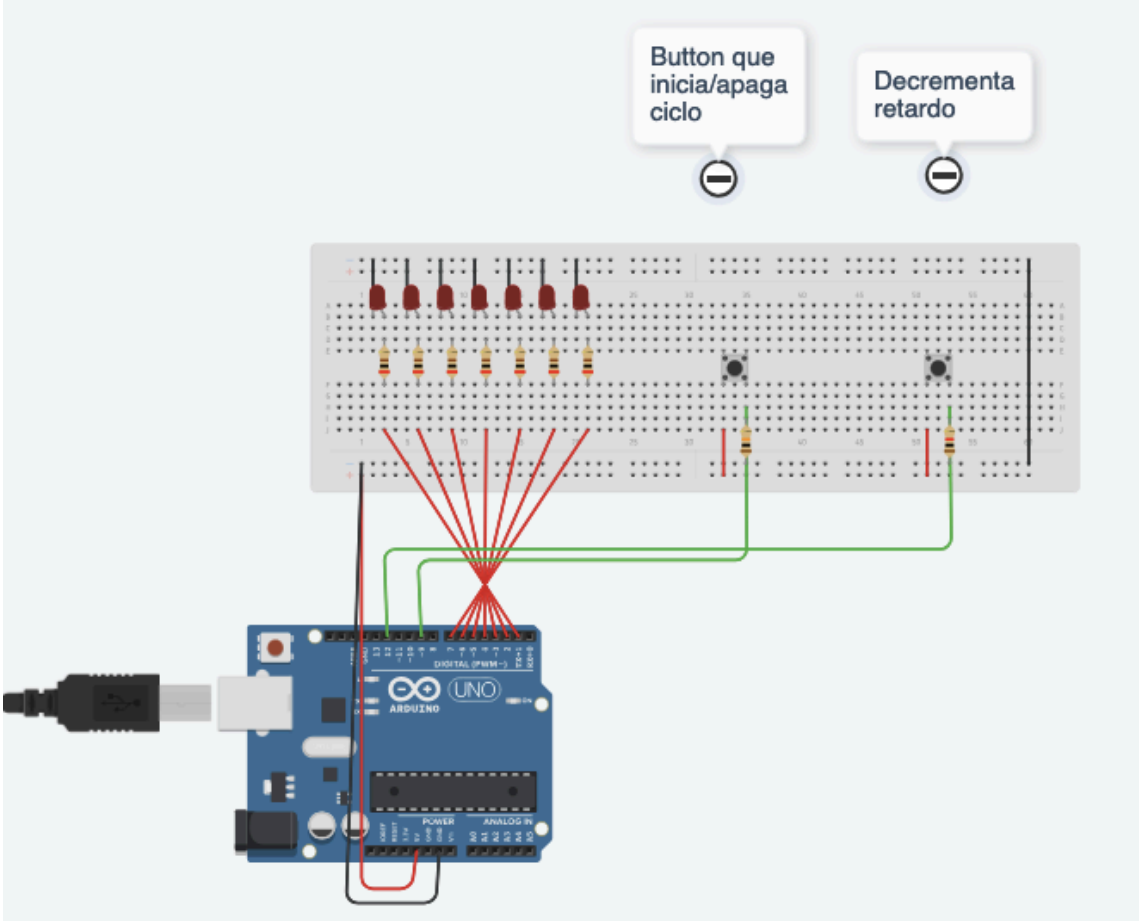
Uno de los principales desafíos fue la implementación del código para controlar la secuencia de LEDs mediante un botón. Aunque la construcción del circuito no presentó complicaciones, el problema surgió al intentar detener la secuencia con una segunda pulsación del botón. Inicialmente, el código detectaba la pulsación para iniciar el ciclo correctamente, pero al colocar la verificación para detener el ciclo al final del código, se generaba un problema. Cuando se detenía la secuencia y el código regresaba al inicio, volvía a detectar el botón presionado y reiniciaba la secuencia inmediatamente, impidiendo un control adecuado sobre el ciclo de encendido y apagado. Se resolvió con tener el código esperando un segundo después de haber presionado el botón dando suficiente tiempo para no encender la secuencia otra vez.

Se identificó una limitación importante al utilizar la función delay() en el código. Dado que delay() es una función bloqueante, al mantener los LEDs encendidos por un tiempo específico, también detiene la ejecución de todo el código. Esto significó que, si durante ese periodo de bloqueo se presionaba el botón destinado a decrementar el retardo, el código no registraba la pulsación, lo que impedía la reducción del tiempo de espera. El equipo consideró la alternativa de utilizar la función millis(), que permite medir el tiempo transcurrido sin bloquear el código, pero no hubo tiempo suficiente para realizar este cambio en el código.

Un desafío no técnico que surgió fue la dificultad para coordinar y ponerse de acuerdo en cómo trabajar juntos como equipo.

Código/Sketch

Versión sin funciones y arreglos:



código:

version1.cpp

```cpp
void setup()
{
    // Entrada
    pinMode(9, INPUT);  // Inicia y termina sequencia
    pinMode(12, INPUT); // Acelera sequencia
    // Salida
    pinMode(1, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(7, OUTPUT);
}

bool inicia = false;

int espera = 1000; // 1000 milisegundos = 1 segundo

bool derecha = true; // si derecha es falso, va a la izquierda

int i = 1;
```

version1.cpp › void loop()

```cpp
void loop()
{
    if(digitalRead(9) == HIGH && inicia == false) inicia = true;
    if (inicia) {
        // se presiono el button?
        if(digitalRead(12) == HIGH) {
            float resta = espera * 0.10;
            espera -= resta;
        }
        digitalWrite(i, HIGH);
        delay(espera);
        digitalWrite(i, LOW);
        if(derecha){
            i += 1;
            // reboto?
            if (i > 6) {
                derecha = false;
                float resta = espera * 0.10;
                espera -= resta;
            }
        } else {
            i -= 1;
            if (i == 1) {
                derecha = true;
                float resta = espera * 0.10;
                espera -= resta;
            }
        }

        if(espera <= 10) {
            derecha = true;
            espera = 1000;
            i = 1;
        }

        if (digitalRead(9) == HIGH) {
            derecha = true;
```

```
53          if(espera <= 10) {
54              derecha = true;
55              espera = 1000;
56              i = 1;
57          }
58
59          if (digitalRead(9) == HIGH) {
60              derecha = true;
61              i = 1;
62              espera = 1000;
63              inicia = false;
64              delay(1000);
65          }
66      }
67  }
```

Version 2 (No se cambio circuito)

version2.cpp

```cpp
const int LEDS[] = {1, 2, 3, 4, 5, 6, 7};
int index = 0;
void setup()
{
    // Entrada
    pinMode(9, INPUT);  // Inicia y termina sequencia
    pinMode(12, INPUT); // Acelera sequencia

    // Salida
    for(int i = 0; i < 7; i++){
        pinMode(LEDS[i], OUTPUT);
    }
}

bool inicia = false;

int espera = 1000; // 1000 milisegundos = 1 segundo

bool derecha = true; // si derecha es falso, va a la izquierda

void loop()
{
    if(digitalRead(9) == HIGH && inicia == false) inicia = true;

    if (inicia) {
        if(digitalRead(12) == HIGH) reducirRetardo();

        digitalWrite(LEDS[index], HIGH);
        delay(espera);
        digitalWrite(LEDS[index], LOW);

        if(derecha) { index += 1; } else { index -= 1; }

        checkRebote();

        if(espera <= 10) {
            resetSequencia();
            inicia = true;
        }
        if (digitalRead(9) == HIGH) resetSequencia();
    }
}
```

```cpp
void loop()
{
    if(digitalRead(9) == HIGH && inicia == false) inicia = true;

    if (inicia) {
        if(digitalRead(12) == HIGH) reducirRetardo();

        digitalWrite(LEDS[index], HIGH);
        delay(espera);
        digitalWrite(LEDS[index], LOW);

        if(derecha) { index += 1; } else { index -= 1; }

        checkRebote();

        if(espera <= 10) {
            resetSequencia();
            inicia = true;
        }
        if (digitalRead(9) == HIGH) resetSequencia();
    }
}
```

Conclusión General

Este proyecto nos permitió aplicar conceptos fundamentales de la computación integrada, como el control de hardware mediante código y la gestión de entradas y salidas en tiempo real. A pesar de los desafíos técnicos, como la gestión de funciones bloqueantes y la estructura del código. Este ejercicio no solo reforzó nuestro entendimiento de los microcontroladores, sino que también nos preparó para abordar proyectos más complejos, como el proyecto final con un Raspberry Pi, donde aplicaremos lo aprendido para desarrollar una solución práctica y útil.

Conclusiones Individuales

Jorge Alberto Morales Reyes

Disfruté esta práctica porque representó un desafío diferente a lo que normalmente estoy acostumbrado. Me pareció muy interesante tener que pensar en cómo el hardware interactúa con el software, lo que amplió mi perspectiva sobre la integración de ambos.

Kevin Isaias Martinez Saucedo

Mi conclusión para esta práctica 1 de secuenciador de leds, es que obtuve conocimientos al usar este programa digital y realizar completamente su diseño y programación en digital fue algo nuevo, al igual que el plantearnos el funcionamiento de este programa, plantear la sintaxis que llevaría los dos, tanto uno básico como uno más avanzado utilizando funciones y arreglos, algo que tendré en cuenta para futuras prácticas a realizar.

Tinkercad

Version 1:

https://www.tinkercad.com/things/kI7djRntz2j-practica1-ver-1-secuencia-de-leds?sharecode=yGXIyQ2Y-_WID0PBeR0MGsUeT1h-iBebHH4K8wyKdoU

Version 2:

https://www.tinkercad.com/things/b4p53lJFsrS-eq-12-practica1-ver-2-secuencia-de-leds?sharecode=irNK4J-8RT-QWHpW53yk1GaezcgPSYbX7P4pPi4bC5g

Video Demostración:

https://youtu.be/H8Q-AJMCIyo

Bibliografía

https://forum.arduino.cc/t/simple-delays-non-blocking-timer-free/635670