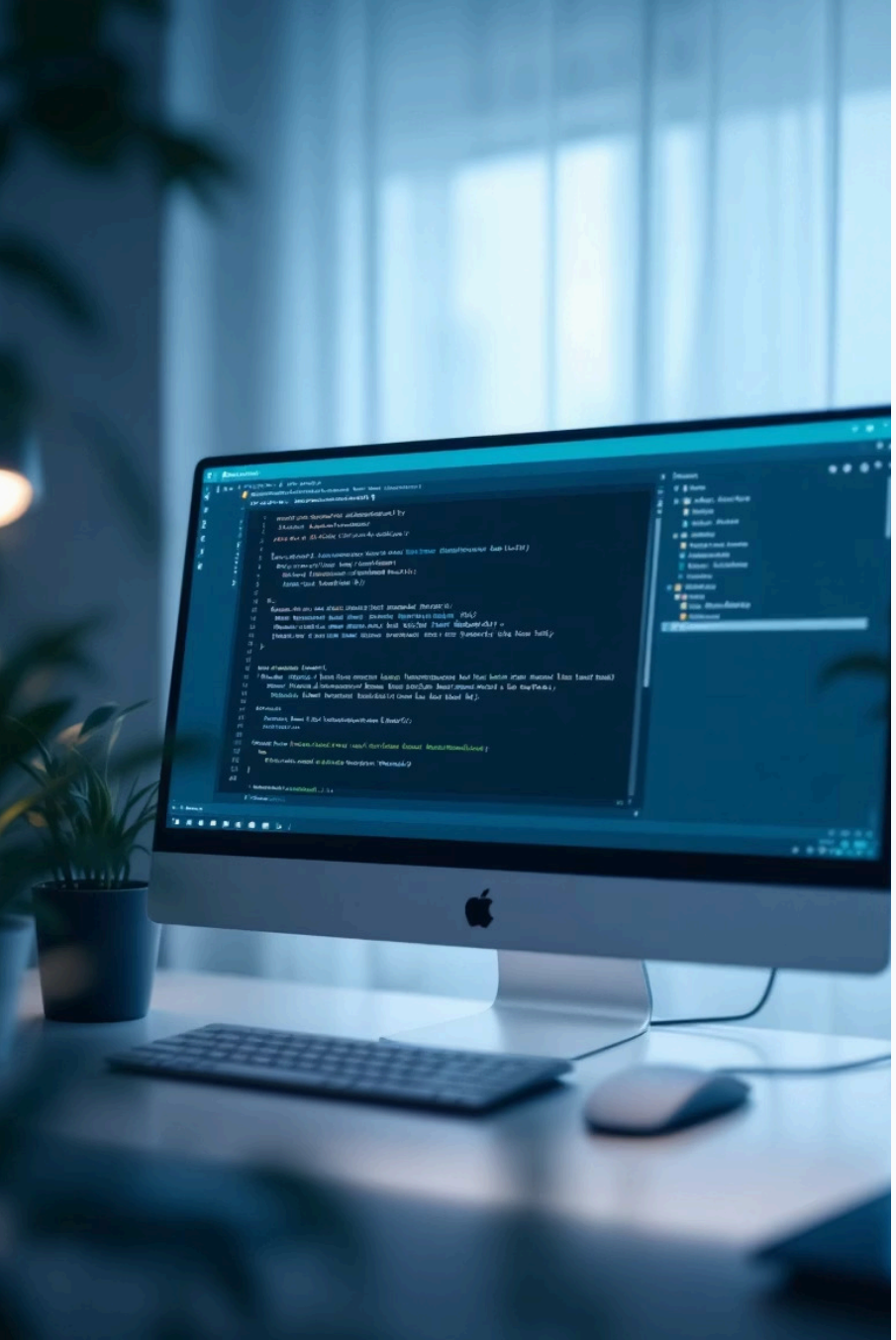


UnoArduSim Documentation Project

Documenting a 4-bit full adder circuit using UnoArduSim V2.9.2 simulator.

Project Overview

This project documents a 4-bit full adder circuit, simulated using UnoArduSim V2.9.2. It provides information for students to understand digital logic and gain hands-on experience with Arduino programming using simulation.

1

Digital Logic

Demonstrate digital logic implementation in embedded systems.

2

Binary Math

Teach binary addition and carry propagation concepts.

3

Hands-on Learning

Provide practical experience with Arduino coding and simulation.

4

Documentation

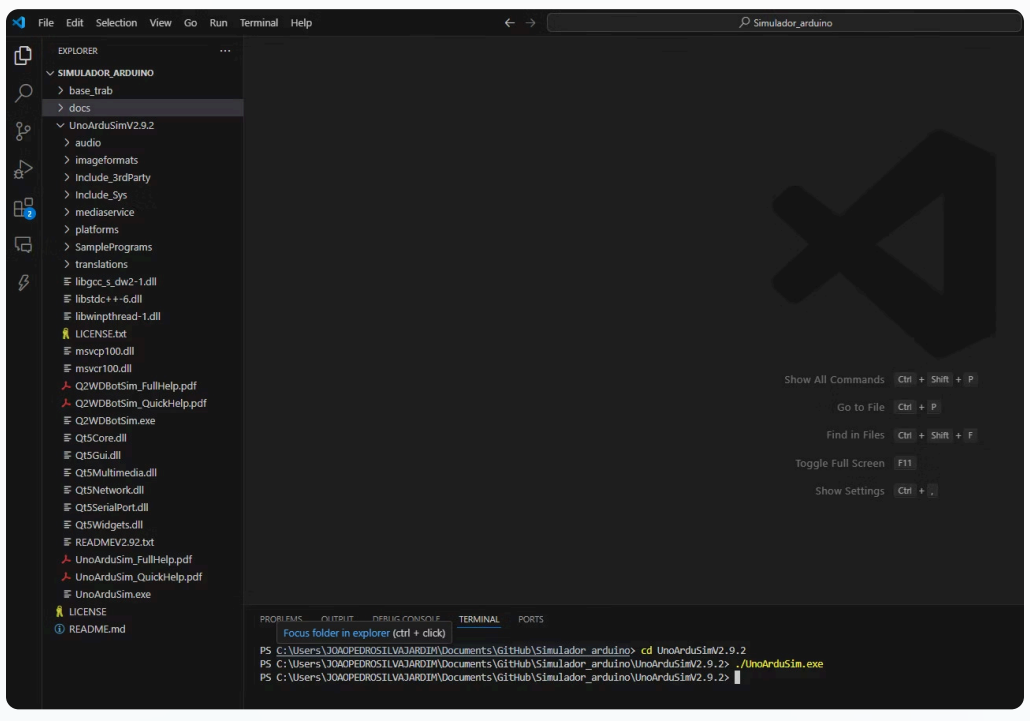
Practice creating clear and effective project documentation.

Code Usage

01

Launch UnoArduSim

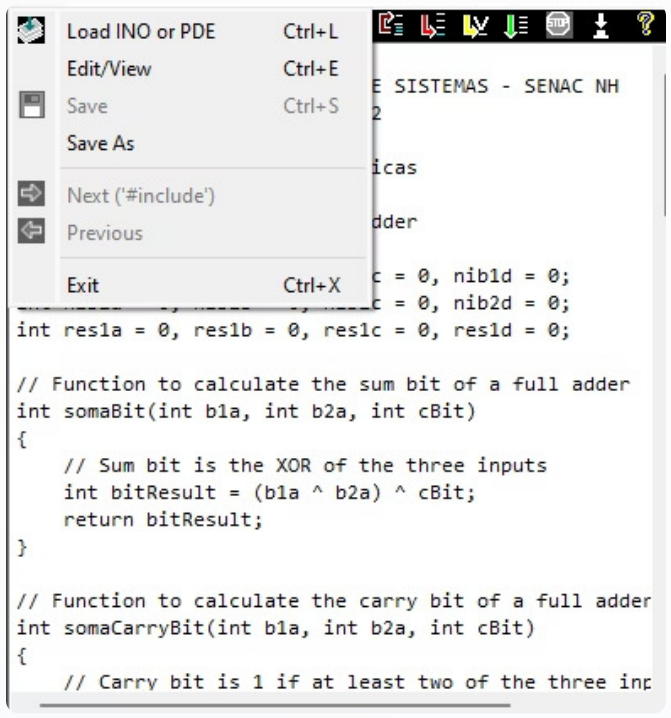
Navigate to UnoArduSimV2.9.2/ folder and run UnoArduSim.exe application.



02

Load Project Sketch

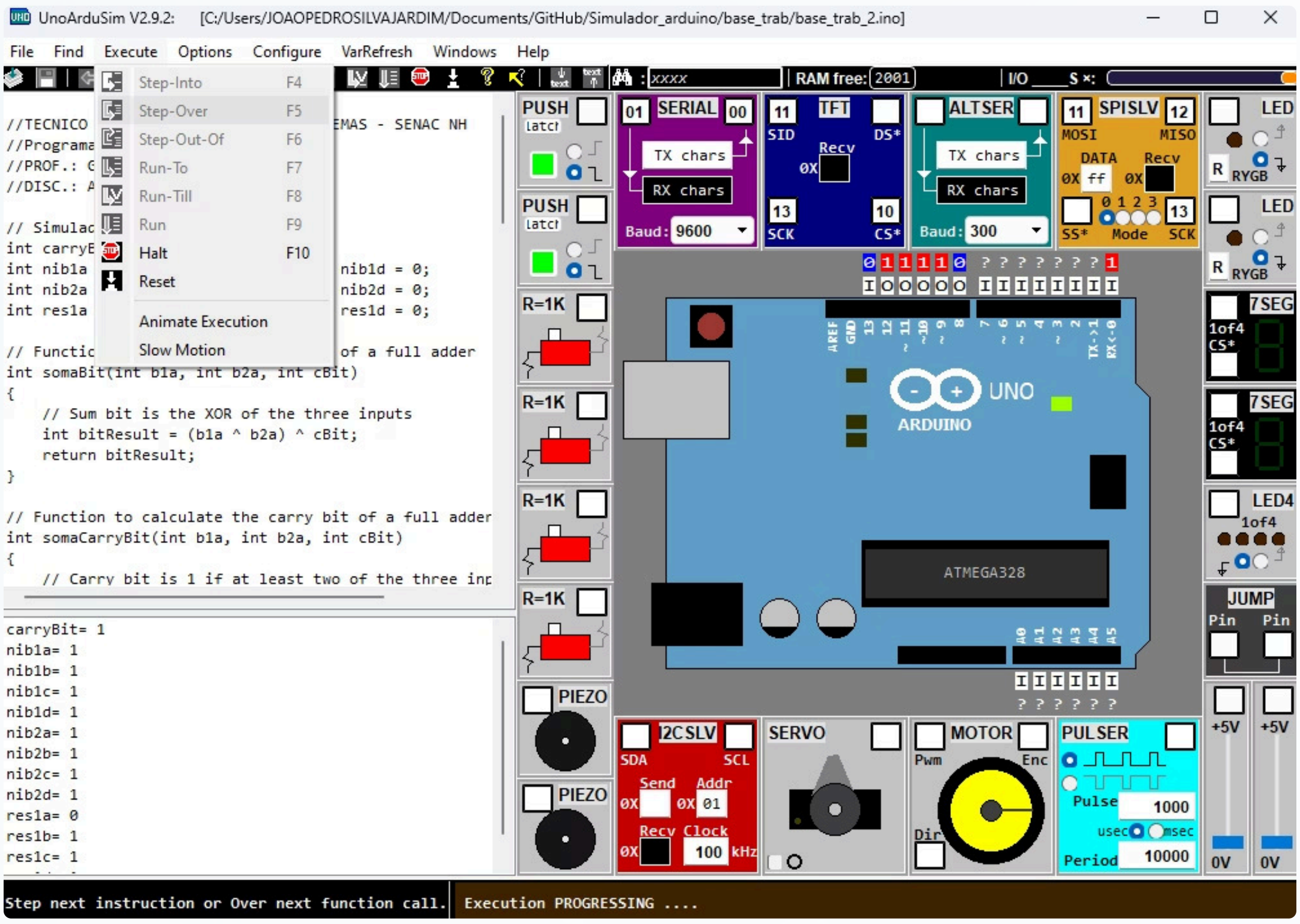
Open File menu and select base_trab/base_trab_2.ino from the project directory.



03

Run Simulation

Click Run button to execute. Observe outputs on pins 8-11 (sum) and pin 12 (carry).



Code Explanation

This Arduino sketch implements a 4-bit full adder using software logic, mimicking digital circuit behavior. It performs binary addition of two 4-bit numbers, handling carry propagation, and is structured for simulation in UnoArduSim.

Key Variables

- **Inputs:** `nib1a-d` (bits of first 4-bit number, pins 0-3), `nib2a-d` (bits of second 4-bit number, pins 4-7).
- **Outputs:** `res1a-d` (4-bit sum, pins 8-11), `carryBit` (final carry, pin 12).
- **Carry Tracking:** `carryBit` propagates between bit positions.

Core Functions

- **somaBit(b1a, b2a, cBit):** Calculates the sum bit for a full adder using XOR logic. Returns 0 or 1.
- **somaCarryBit(b1a, b2a, cBit):** Determines the carry-out bit using OR of AND pairs logic. Returns 0 or 1.

Setup & Loop

- **Setup Function:** Configures pins 0-7 as INPUTs for numbers, and pins 8-12 as OUTPUTs for sum and carry.
- **Loop Function:** Reads inputs, performs chained bit-by-bit addition, and writes the 4-bit sum and final carry to output pins.

Bugs Fixed in Original Code

Variable Initialization

Corrected improper initialization where only the last variable in declaration lists was set to 0, causing undefined behavior.

Logic Optimization

Improved inefficient logic in somaBit and somaCarryBit functions using proper XOR and AND operations for better performance.

Code Cleanup

Removed redundant always-true conditions and added clear English comments for better code maintainability and understanding.

Below are visual examples of the key code improvements and bug fixes implemented:

```
// Simulador de circuito FullAdder
int carryBit = 0;
int nib1a = 0, nib1b = 0, nib1c = 0, nib1d = 0;
int nib2a = 0, nib2b = 0, nib2c = 0, nib2d = 0;
int res1a = 0, res1b = 0, res1c = 0, res1d = 0;
```

```
// Function to calculate the sum bit of a full adder
int somaBit(int b1a, int b2a, int cBit)
{
    // Sum bit is the XOR of the three inputs
    int bitResult = (b1a ^ b2a) ^ cBit;
    return bitResult;
}

// Function to calculate the carry bit of a full adder
int somaCarryBit(int b1a, int b2a, int cBit)
{
    // Carry bit is 1 if at least two of the three inputs are 1
    int newCarry = (b1a && b2a) || (b1a && cBit) || (b2a && cBit);
    return newCarry;
}
```

```
// Main loop: Reads inputs, performs addition, writes outputs
void loop()
{
    // Read input bits from pins
    nib1a = digitalRead(0);
    nib1b = digitalRead(1);
    nib1c = digitalRead(2);
    nib1d = digitalRead(3);
    nib2a = digitalRead(4);
    nib2b = digitalRead(5);
    nib2c = digitalRead(6);
    nib2d = digitalRead(7);

    // Perform 4-bit addition using full adder logic
    carryBit = 0;
    res1a = somaBit(nib1a, nib2a, carryBit);
    carryBit = somaCarryBit(nib1a, nib2a, carryBit);
    res1b = somaBit(nib1b, nib2b, carryBit);
    carryBit = somaCarryBit(nib1b, nib2b, carryBit);
    res1c = somaBit(nib1c, nib2c, carryBit);
    carryBit = somaCarryBit(nib1c, nib2c, carryBit);
    res1d = somaBit(nib1d, nib2d, carryBit);
    carryBit = somaCarryBit(nib1d, nib2d, carryBit);

    // Write result bits to output pins
    digitalWrite(8, res1a);
    digitalWrite(9, res1b);
    digitalWrite(10, res1c);
    digitalWrite(11, res1d);
    digitalWrite(12, carryBit);
}
```

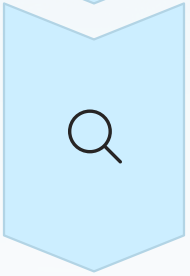
Future Improvements To Fix

Expand Bit Width

Extend functionality to 8-bit or 16-bit adders for handling larger numbers and more complex calculations.

Performance Optimization

Use bitwise operations and input validation to improve execution speed and error handling capabilities.



Modular Design

Restructure code for modularity and reusability in other projects, with enhanced documentation and visual diagrams.

Key Learning Outcomes

Digital Logic

Understanding binary mathematics, logic gates, and carry propagation in digital circuits through practical implementation and simulation.

Arduino Programming

Experience with embedded systems programming, pin configuration, and digital input/output operations.

Technical Documentation

Developing professional documentation skills for software projects.

Debugging & Optimization

Identifying code issues, implementing fixes, and optimizing functions.

Project License

This UnoArduSim documentation project, is released under the MIT License. This permissive open-source license allows for broad usage and collaboration while protecting the original authors.