دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

پروژه ساختار و زبان کامپیوتر
مهندسی کامپیوتر

پروژه اول: شبیه ساز مدارات منطقی

نگارش

پارسا نوروزی منش

آرش قوامی

هلنا دهخوارقانیان

امیرمحمد نصراله نژاد

استاد راهنما

استاد اسدی

دی ۱۴۰۳

چکیده

در این پروژه، هدف اصلی بررسی و پیاده‌سازی منطق قابل پیکربندی مجدد است که به ما امکان می‌دهد سخت‌افزار دیجیتال را به صورت پویا بازآرایی کنیم و عملکرد سیستم را بر اساس نیازهای خاص بهینه‌سازی نماییم. این پروژه به طور ویژه بر طراحی و توسعه نرم‌افزاری برای شبیه‌سازی مدارهای منطقی و همچنین پیاده‌سازی سخت‌افزار مبتنی بر برد آردوینو تمرکز دارد.

در بخش نرم‌افزاری، یک شبیه‌ساز با رابط کاربری گرافیکی طراحی می‌شود که به کاربر اجازه می‌دهد جداول صحت را برای مدارهای منطقی تعریف کرده و آن‌ها را آزمایش کند. همچنین این نرم‌افزار قادر است داده‌های تعریف‌شده را از طریق ارتباط سریال به برد آردوینو ارسال کند.

در بخش سخت‌افزاری، برد آردوینو با اجزای ورودی و خروجی مانند دیپ‌سوییچ‌ها و ال‌ای‌دی‌ها پیکربندی شده و یک مدار نمونه مانند جمع‌کننده دو بیتی پیاده‌سازی می‌شود. ورودی‌های مدار از طریق دیپ‌سوییچ تعیین شده و خروجی‌ها به صورت بصری روی ال‌ای‌دی‌ها نمایش داده می‌شوند.

این پروژه با هدف تقویت توانایی‌های دانشجویان در طراحی و پیاده‌سازی مدارهای منطقی دیجیتال، برقراری ارتباط میان نرم‌افزار و سخت‌افزار و آشنایی عملی با فناوری‌های پیکربندی مجدد انجام می‌شود. نتیجه نهایی سیستمی خواهد بود که امکان شبیه‌سازی و اجرای زنده مدارهای منطقی را با کارایی و انعطاف‌پذیری بالا فراهم می‌آورد.

فهرست مطالب

فهرست تصاویر

فصل ۱

مقدمه

این پروژه با هدف طراحی و شبیه‌سازی یک سیستم پویا و هوشمند برای اجرای منطق دیجیتال قابل پیکربندی مجدد آغاز می‌شود. ایده اصلی این است که بتوان ساختارهای دیجیتال را به صورت انعطاف‌پذیر و متناسب با نیازهای خاص تغییر داد، بدون نیاز به تغییرات فیزیکی در سخت‌افزار. در این راستا، از شبیه‌سازی نرم‌افزاری به جای برد واقعی استفاده می‌شود که این خود چالشی جذاب و در عین حال کاربردی است.

ابتدا، پروژه با بررسی مبانی نظری منطق قابل پیکربندی مجدد و اهمیت آن در سیستم‌های دیجیتال شروع می‌شود. این مرحله شامل مطالعه کاربردهای این فناوری و توانایی آن در بهبود انعطاف‌پذیری و کارایی سیستم‌ها است.

در گام بعدی، طراحی نرم‌افزاری انجام می‌شود که به کاربر اجازه می‌دهد جداول منطقی مختلف را تعریف و شبیه‌سازی کند. این نرم‌افزار باید رابط کاربری ساده و قابل فهمی داشته باشد تا بتوان ورودی‌های مدارات منطقی را تعریف و خروجی‌های آن‌ها را به راحتی مشاهده کرد.

سپس، یک مدار نمونه مانند جمع‌کننده دو بیتی طراحی و پیاده‌سازی می‌شود. این مدار به گونه‌ای شبیه‌سازی می‌شود که بتوان ورودی‌ها را از طریق نرم‌افزار به آن داد و خروجی‌ها را به صورت روشن شدن چراغ‌ها یا سایر نمایشگرها مشاهده کرد. این مرحله، ترکیبی از طراحی نرم‌افزاری و تحلیل مدارهای منطقی است.

در ادامه، نتایج شبیه‌سازی و عملکرد مدار از جنبه‌های مختلف مانند دقت، سرعت و کارایی بررسی می‌شود. این تحلیل‌ها مشخص می‌کنند که آیا سیستم طراحی شده توانسته اهداف تعریف شده را به درستی محقق کند یا خیر.

فصل ۲

رابط گرافیکی

این کد به زبان پایتون و با استفاده از کتابخانه‌های tkinter برای رابط گرافیکی و pandas برای کار با داده‌ها نوشته شده است. هدف این برنامه مدیریت جداول منطقی و ارسال داده‌ها به دستگاه‌های خارجی مانند آردوینو است. در اینجا گزارش جامعی از عملکرد کد آورده شده است:

۱-۲ عملکرد برنامه مدیریت جداول با Pandas و Tkinter

این برنامه با استفاده از رابط گرافیکی Tkinter به کاربران این امکان را می‌دهد که جداول منطقی شامل ورودی‌ها و خروجی‌ها را ایجاد، ویرایش، مشاهده، و حتی به آردوینو ارسال کنند. با استفاده از این برنامه، کاربر قادر است تعداد ورودی‌ها و خروجی‌ها را مشخص کرده، مقادیر مربوط به هر ترکیب ورودی را وارد کند و سپس این جداول را برای استفاده‌های بعدی ذخیره و مدیریت کند. در نهایت، دنباله‌های تولیدی به پورت سریال آردوینو ارسال می‌شوند.

۲-۲ ویژگی ها و عملکرد برنامه

۱-۲-۲ ایجاد جدول جدید

کاربر قادر است نام یک جدول جدید وارد کند. سپس تعداد ورودی‌ها و خروجی‌ها را مشخص می‌کند. برای هر ورودی و خروجی، نام مناسب از کاربر درخواست می‌شود. بعد از آن، ترکیب‌های مختلف ورودی به صورت باینری تولید شده و از کاربر خواسته می‌شود که مقدار خروجی را برای هر ترکیب ورودی وارد کند. برای هر ترکیب ورودی، برنامه مقادیر خروجی‌ها را دریافت کرده و آن‌ها را به صورت عدد دهدهی ذخیره

می‌کند. جدول نهایی به همراه دنباله مربوطه در حافظه برنامه ذخیره می‌شود.

۲-۲-۲ نمایش جدول

کاربر می‌تواند جداول ذخیره شده را مشاهده کند. برای این کار، از کاربر خواسته می‌شود که نام جدول مورد نظر را وارد کند. پس از انتخاب جدول، داده‌ها به صورت گرافیکی و با استفاده از یک پنجره جدید به کاربر نمایش داده می‌شوند. ورودی‌ها با رنگ آبی و خروجی‌ها با رنگ سبز نشان داده می‌شوند.

۲-۲-۳ حذف جدول

کاربر می‌تواند جداول ذخیره شده را حذف کند. پس از انتخاب نام جدول، جدول و دنباله مربوطه از حافظه برنامه حذف می‌شوند.

۲-۲-۴ ارسال داده ها به آردوینو

کاربر می‌تواند دنباله‌های تولید شده را به آردوینو ارسال کند. برای این کار، کاربر باید نام جدول مورد نظر را وارد کند. دنباله مربوطه به صورت رشته‌ای به پورت سریال آردوینو ارسال می‌شود و آردوینو قادر خواهد بود داده‌ها را دریافت کند.

۲-۳ نحوه عملکرد حدوال و دنباله ها

۲-۳-۱ ساختار جداول

هر جدول شامل تعدادی ورودی و خروجی است. ورودی‌ها و خروجی‌ها در قالب ستون‌ها در یک DataFrame ذخیره می‌شوند. ترکیب‌های مختلف ورودی‌ها به صورت باینری ساخته شده و از کاربر خواسته می‌شود که مقادیر خروجی‌ها را برای هر ترکیب ورودی وارد کند. این مقادیر خروجی به صورت عدد باینری ذخیره شده و به عدد دهدهی تبدیل می‌شوند.

۲-۳-۲ ذخیره دنباله ها

دنباله‌ای که شامل اطلاعات جدول (نام جدول، تعداد ورودی‌ها، تعداد خروجی‌ها و مقادیر خروجی‌ها) است، ذخیره می‌شود. این دنباله به صورت یک لیست در دیکشنری sequences ذخیره می‌شود.

۲-۳-۳ ارسال دنباله به آردوینو

دنباله‌ها از طریق پورت سریال به آردوینو ارسال می‌شوند. آردوینو این داده‌ها را دریافت کرده و می‌تواند آن‌ها را برای اعمال مختلف استفاده کند.

۲-۴ جزییات فنی

۲-۴-۱ کتابخانه های استفاده شده

tkinter برای ایجاد رابط گرافیکی
pandas برای مدیریت و پردازش داده ها به صورت جدول
itertools برای تولید مقادیر و حالت های ممکن باینری
serial برای ارتباط با آردوینو و ارسال داده به آن

۲-۴-۲ توابع ورودی

تابع ask-for-integer

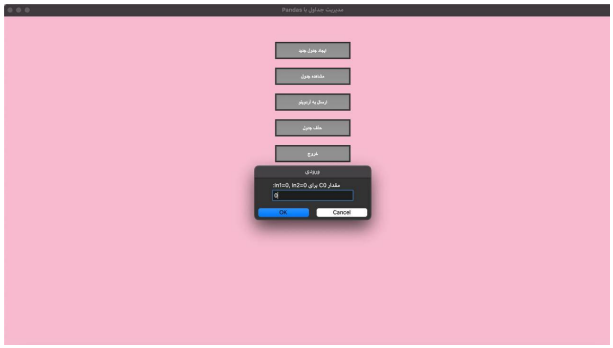
دریافت ورودی عددی از کاربر با استفاده از simpledialog. اگر کاربر ورودی نامعتبر وارد کند، پیام خطا نشان داده می‌شود.

تابع ask-for-binary

دریافت مقدار باینری (۰ یا ۱). بررسی می‌کند که کاربر فقط مقادیر مجاز وارد کند و در غیر این صورت خطا می‌دهد.

تابع create-new-table

کاربر یک نام برای جدول انتخاب می‌کند. تعداد ورودی‌ها و خروجی‌ها را مشخص می‌کند. نام ورودی‌ها و خروجی‌ها به صورت جداگانه از کاربر دریافت می‌شود. ترکیب تمام حالت‌های ممکن برای ورودی‌ها (۰ و ۱) با استفاده از itertools.product تولید می‌شود. مقادیر خروجی برای هر ترکیب توسط کاربر وارد می‌شود. مقادیر خروجی به صورت باینری جمع شده و سپس به عدد دهدهی تبدیل می‌شود. جدول در tables ذخیره شده و دنباله مربوطه در sequences ذخیره می‌شود. اگر نام جدول تکراری باشد، خطا نمایش داده می‌شود.



شکل ۲-۱: تعیین مقادیر خروجی بر اساس ورودی

شکل ۲-۲: ساختن جدول جدید

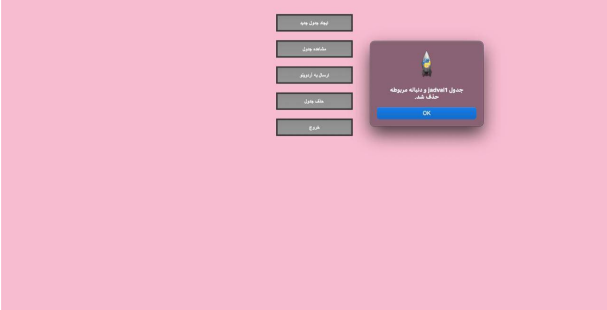
تابع send-to-arduino

نام یک جدول از کاربر درخواست می‌شود. دنباله مربوط به جدول انتخابی از sequences گرفته می‌شود.

تابع view-table

لیست جداول موجود نمایش داده می‌شود. جدول انتخابی کاربر در یک پنجره جدید به کمک Canvas نمایش داده می‌شود. ستون‌ها (ورودی‌ها و خروجی‌ها) و مقادیر جدول در Canvas به صورت گرافیکی رسم می‌شوند.

شکل ۲-۳: مشاهده جدول ساخته شده



شکل ۲-۴: حذف کردن جدول مورد نظر

تابع delete-table

جدولی که نامش توسط کاربر وارد می‌شود از هر دو tables و sequences حذف می‌شود. اگر جدول مورد نظر موجود نباشد، خطا نمایش داده می‌شود.

فصل ۳

شبیه‌سازی مدار و برنامه نویسی برد آردوینو

باید کدی به زبان آردوینو بنویسیم که بورد آردوینو را برنامه‌ریزی کند و امکان انتقال جداول طراحی شده در رابطه گرافیکی از طریق آن به بورد فراهم شود تا بتوانیم با تعیین ورودی‌های بورد با استفاده از dip switch، خروجی‌های مدار را به صورت LED مشاهده کنیم.
طبق صورت پروژه، تنها قابلیت شبیه‌سازی مدار روی بورد کافی نیست و باید نرم‌افزار ما (که تا به حال تنها نقش رابط گرافیکی برای طراحی جداول درستی توسط کاربر را داشت) هم باید به تنهایی بتواند مدار را شبیه‌سازی کند. بنابراین ابتدا به سراغ شبیه‌سازی مدار تنها توسط نرم‌افزار نوشته شده و بدون بورد و کد آردوینو می‌رویم.

۳-۱ شبیه‌سازی مدار با نرم‌افزار نوشته شده و بدون بورد و کد آردوینو

نرم‌افزار ما تا به این جای کار تنها یک رابط گرافیکی برای طراحی جدول بود. حال برای این‌که قابلیت گرفتن ورودی‌های مدار از کاربر و شبیه‌سازی آن‌ها را هم پیدا کند نیاز به ارتباط بین کاربر و برنامه از طریق ترمینال هم داریم. برای این کار از مکانیزم multithreading استفاده می‌کنیم. به این معنا که کدی که مسئول هندل کردن پنجره‌های tkinter و pandas در رابط گرافیکی بود را در یک thread و کد مربوط به رابط ترمینالی را در یک thread دیگر تعریف کرده و آن‌ها را اجرا می‌کنیم تا همزمان اجرا شوند. در مورد رابط گرافیکی در بخش مربوطه توضیح داده شده است. حال به توضیح رابط ترمینالی می‌پردازیم و بخش‌های مختلف پیاده‌سازی آن و نحوه‌ی کار با آن را شرح می‌دهیم.

۳-۱-۱ تابع terminal-input()

در این تابع یک حلقه‌ی بی‌نهایت تعریف شده است که از کاربر در ترمینال ورودی می‌گیرد. ورودی‌های ممکن از طرف کاربر واژه‌های process و define هستند و در صورت وارد کردن دستور دیگر پیغام خطا می‌دهد. با ورودی گرفتن کلمه‌ی define تابع define-curcuit-input() برای مشخص شدن یک ورودی توسط کاربر اجرا می‌شود. با گرفتن کلمه‌ی process هم تابع process-curcuit() برای پردازش و محاسبه‌ی مقادیر متغیرها و خروجی‌های مدار اجرا می‌شود.

۳-۱-۲ تابع define-curcuit-input()

پس از اجرا شدن این تابع ابتدا از کاربر نام ورودی‌ای که می‌خواهد تعیین کند پرسیده می‌شود. کاربر حق دارد یکی از مقادیر in۰ تا in۷ را ورودی دهد و سپس مقدار ۰ یا ۱ را به عنوان مقدار آن ورودی تعیین کند در غیراین‌صورت با خطا روبه‌رو می‌شود. در صورت ورودی دادن صحیح این مقدار در دیکشنری curcuit-inputs ذخیره می‌شود.

۳-۱-۳ تابع process-curcuit()

در این تابع قرار است پردازش مربوط به مقدار متغیرها و خروجی‌ها در مدار انجام شود. ابتدا نام همه‌ی متغیرهای موجود در مدار از روی جدول‌های طراحی شده توسط کاربر خوانده شده در یک دیکشنری با value برابر با مقدار آن‌ها قرار می‌گیرند که در ابتدا این مقدار برای متغیرهای غیرورودی برابر -۱ و برای ورودی‌های مدار طبق مقادیر تعیین شده است. سپس روی همه‌ی این متغیرها پیمایش انجام می‌شود و با تابع calculate-curcuit-variable() مقدار همه‌ی آن‌ها یکی یکی حساب می‌شود.

توضیح بیشتر در مورد تابع calculate-curcuit-variable()

این تابع به صورت بازگشتی عمل می‌کند. ابتدا جدولی که طبق آن نحوه‌ی محاسبه‌ی متغیری که قصد محاسبه‌ی آن را داریم تعیین شده است با پیمایش روی جدول‌ها پیدا می‌شود. سپس مقادیر متغیرهای ورودی در آن جدول به صورت بازگشتی محاسبه شده و تعیین می‌شود که متغیری که می‌خواهیم مقدارش را پیدا کنیم باید از روی کدام سطر جدول تعیین شود.

in0	in1	c
0	0	0
0	1	1
1	0	1
1	1	0

شکل ۳-۱: جدول مثال ۱

c	in2	led0
0	0	0
0	1	1
1	0	1
1	1	0

شکل ۳-۲: جدول مثال ۲

۳-۱-۴ مثال از نحوه‌ی کار با رابط ترمینالی

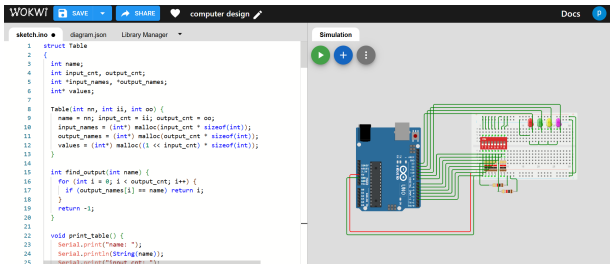
فرض کنید می‌خواهیم با دو جدول مداری طراحی کنیم که xor سه ورودی in۰، in۱ و in۲ را محاسبه کند. برای این کار یکی متغیر میانی مانند c را برابر in۱ xor in۰ در یک جدول تعریف کرده و سپس در جدول بعدی led۰ یعنی خروجی مدار را برابر in۲ xor c تعریف می‌کنیم. جدول‌های طراحی شده: حال با استفاده از دستور define مقدار ورودی‌ها را به صورت دلخواه تعیین می‌کنیم و سپس با دستور process مقادیر متغیرها و خروجی را می‌بینیم.

۳-۲ شبیه‌سازی مدار با بورد و کد آردوینو

حال به سراغ شبیه‌سازی مدار با کد و بورد آردوینو می‌رویم. از آن‌جایی که بورد فیزیکی در اختیار نداریم، برای شبیه‌سازی بورد آردوینو از شبیه‌ساز wokwi استفاده می‌کنیم. به این صورت که ابتدا مدار موردنیاز روی بورد را در محیط گرافیکی می‌بندیم و سپس با کمی زدن کد در بخش ترمینال آن می‌توانیم خروجی آن را به صورت زیپ دانلود کرده و در Code VS اجرا کنیم.

```
enter the command please: define
enter the curcuit input: in0
enter the value: 1
enter the command please: define
enter the curcuit input: in1
enter the value: 0
enter the command please: define
enter the curcuit input: in2
enter the value: 1
enter the command please: process
----------
in0: 1
in1: 0
in2: 1
----------
c: 1
led0: 0
```

شکل ۳-۳: تعیین کردن ورودی‌ها و سپس دیدن نتیجه



شکل ۳-۴: تصویری از محیط شبیه‌ساز wokwi

۳-۲-۱ توضیحاتی در مورد مدار بسته شده روی بورد

در این مدار از switch dip هشت‌تایی برای تعیین ورودی‌های in۰ تا Vin استفاده شده و مقادیر ورودی از GPIO های ۲ تا ۹ قابل خواندن هستند. همچنین ۴ LED برای نمایش خروجی‌های مدار هم وجود دارد که برای تعیین روشن یا خاموش بودن آن‌ها می‌توان از GPIO های ۱۰ تا۱۳ استفاده کرد.

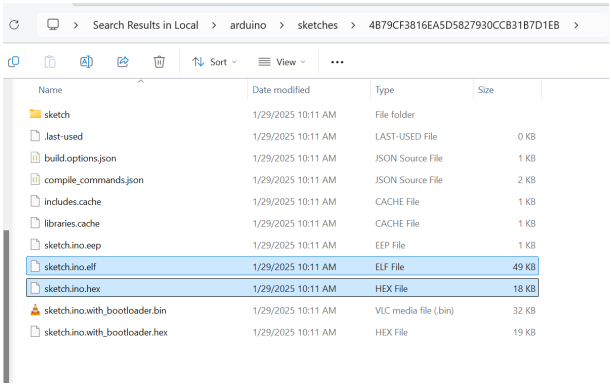
۳-۲-۲ نحوه‌ی اجرا روی Code VS

باید اکستنشن Simulator Wokwi را روی Code VS نصب کنیم. پس از باز کردن زیپ گرفته شده از سایت wokwi ابتدا باید فایل wokwi.toml را به این صورت در همان فولدر بسازیم. پس از آن باید فایل sketch.ino را با IDE Arduino کامپایل کنیم. سپس باید دو فایل از path ای که sketch های آردوینو

wokwi.toml

```
[wokwi]
rfc2217ServerPort = 4000
version = 1
firmware = 'sketch.ino.hex'
elf = 'sketch.ino.elf'
```

شکل ۳-۵: محتوای فایل wokwi.toml



شکل ۳-۶: فایل‌های موردنیاز ایجاد شده پس از کامپیال توسط IDE Arduino

در آن ذخیره می‌شوند به درون فولدر پروژه کپی کنیم و آن دو فایل sketch.ino.elf و sketch.ino.hex هستند. در نهایت هم با کلیدهای Ctrl + shift + p و کلیک بر :wokwi simulator start شبیه‌سازی آغاز می‌شود.

۳-۲-۳ انتقال جدول‌ها از رابط گرافیکی به آردوینو

برای برقراری ارتباط بین آردوینو و رابط گرافیکی، از طریق پورت serial بورد و همچنین رابط گرافیکی به پورت ۴۰۰۰ localhost متصل می‌شویم و با ()Serial.print بین آن‌ها جدول‌ها را به صورت رشته‌هایی متشکل از نام جدول، تعداد ورودی، تعداد خروجی و مقادیر خروجی به ازای حالات مختلف ورودی منتقل می‌کنیم. البته لازم به ذکر است که به دلیل حافظه‌ی کم بورد آردوینو تمام نام‌ها اعم از اسم متغیرهای ورودی و خروجی و همچنین اسم جدول‌ها را به اعداد مپ کردیم و به صورت عدد به آردوینو منتقل کردیم تا حافظه‌ی کمتری اشغال شود.

in0	in2	led0	c0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

شکل ۳-۷: یک جدول مثال طراحی شده در رابط گرافیکی

```
--------
new table added!
name: 1
input_cnt: 2
output_cnt: 2
inputs: 2 4
outputs: 10 14
values: 0 2 2 1
--------
```

شکل ۳-۸: جدول پس از ارسال به آردوینو و دریافت شدن توسط آردوینو

۳-۲-۴ نحوه‌ی پیاده‌سازی کد آردوینو

در تابع ()loop که مدام اجرا می‌شود در صورتی که ورودی‌ای با استفاده از switch dip تغییر کند یا از طریق پورت سریال جدول جدیدی به آردوینو ارسال شود تابع ()process-circuit اجرا شده و پس از پردازش مقدار متغیرها و خروجی‌های مدار را محاسبه کرده و در نهایت مقادیر خروجی را در قالب LED ها نشان می‌دهد.

اندکی توضیح در مورد تابع ()process-circuit

پیاده‌سازی این تابع کاملا مشابه تابع مشابه برای پردازش و محاسبه‌ی مقادیر متغیرها در نرم‌افزار است وقتی نیاز داشتیم مقادیر را بدون آردوینو محاسبه و مدار را تنها در نرم‌افزار شبیه‌سازی کنیم. در این تابع هم ابتدا همه‌ی متغیرهای مدار از روی جدول‌ها شناسایی می‌شوند. سپس در تابع بازگشتی calculate-

in0	in2	led0	c0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

شکل ۳-۹: طراحی جدول ۱

in1	in3	c0	led1	led2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

شکل ۳-۱۰: طراحی جدول ۲

variable-value() برای محاسبه‌ی مقدار یک متغیر، ابتدا جدولی که نحوه‌ی محاسبه‌ی آن را تعیین می‌کند پیدا می‌شود. سپس مقادیر ورودی آن جدول به طور بازگشتی محاسبه می‌شوند و به این صورت مقدار آن متغیر تعیین می‌شود.

۳-۲-۵ پیاده‌سازی مثال جمع‌کننده‌ی دوبیتی با استفاده از نرم‌افزار و آردوینو

ابتدا جدول‌های موردنظر را طراحی کرده و و به آردوینو می‌فرستیم. سپس پردازش را به ازای ورودی in۰=۱، in۱=۱، in۲=۰، in۳=۱ انجام می‌دهیم که معادل جمع ۳ + ۲ است و باید در نهایت مقدار ۵ یعنی led۲=۱، led۱=۰، led۰=۱ و رقم نقلی c۰=۰ به دست آید.

```
new table added!
name: 1
input_cnt: 2
output_cnt: 2
inputs: 2 4
outputs: 10 14
values: 0 2 2 1
```

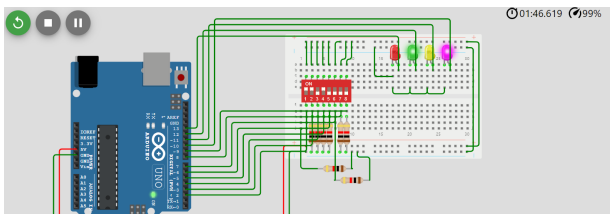
شکل ۳-۱۱: دریافت جدول ۱

```
new table added!
name: 2
input_cnt: 3
output_cnt: 2
inputs: 3 5 14
outputs: 11 12
values: 0 2 2 1 2 1 1 3
```

شکل ۳-۱۲: دریافت جدول ۲

```
enter the command please: define
enter the curcuit input: in0
enter the value: 1
enter the command please: define
enter the curcuit input: in1
enter the value: 1
enter the command please: define
enter the curcuit input: in2
enter the value: 0
enter the command please: define
enter the curcuit input: in3
enter the value: 1
enter the command please: process
----------
in0: 1
in1: 1
in2: 0
in3: 1
----------
led0: 1
c0: 0
led1: 0
led2: 1
enter the command please:
```

شکل ۳-۱۳: پرداش با نرم‌افزار



شکل ۳-۱۴: پردازش با آردوینو