



University of Wales, Trinity Saint David

Gower College Swansea, Tycoch College

Designing an Affordable Means of Entertainment – Initial Report

Ronan Pesarillo

PES15826963

Higher National Diploma (HND) in Electrical and Electronic Engineering

Monday, 12 June 2023

Group Project and Research Methods

Contents

Abstract	2
Introduction	2
Organisation	3
Project Plan	4
Research in the chosen EWB Challenge Design	6
Project-Relevant Areas, Subjects and Technologies (Research)	6
Calculation of Costs – including Bill of Materials (BoM)	7
References	8

Abstract

'Project' is where students are assigned to aim at developing a project whilst maintaining the execution of realistic objectives. All while writing down the essence of the product and how it follows its requested theme.

The document's aim is about producing a project and writing down whether the project meets the standards and queries provided by the assignment briefs and providing brief quantities of evidence where due.

The year's assignment inquires the production of a project that is both involved with 'Engineers Without Borders' and of relatively simplistic and entertaining design so that it engagingly educates those in developing countries about the topics of STEM (science, technology, engineering and mathematics). ✓

Introduction

When an assignment requires a team, i.e: two or more people and applies emphasis in group co-ordination, teamwork, cooperation, etc., often in such situations that other students will be involved in small groups. Each student in a group deal with specific roles usually designated by a team leader (see 'Organisation' for examples). Each student will tackle specific roles to reduce expected individual effort and save on development time. On the other hand, teamwork may lead to obstructive events such as: typical group arguments, sketchily devised breaks, temperamental disfigurement from group absentees, etc. — Not relevant.

However, with the quantity of students involved in the college year's assignment, the way of working solo is the only method to complete the aim set out by the assignment. To produce a working project within a set deadline. To play every possible role that is involved in theme of the assignment, leading to larger accumulations of psychological obstruction. To present every facet of action performed throughout.

In 'Project', their aim is to design a STEM project for school children in developing countries. The project must be of easy-to-medium difficulty and include step-by-step instructions. And the project must be set to encourage and promote STEM in an engaging way, with fun and interactivity in project's priority after development. ✓

This section could be made smaller

wording

Organisation

The organisation of the project only involves Ronan Pesarillo as part of the development 'team' for a lack of a fitting word. *✓*

Roles for the projects mostly include, but not limited to: 'Human Resources Manager', 'Report Writing', '3D Modelling', 'Prototype Illustrator', 'Architectural Technologist', 'Graphic Designer', 'Product Buyer', 'Budget Manager', 'Technical Supervisor', 'Software Developer', 'Solderer', etc.

But said roles may not fit verbatim for comprehension reasons and must be reworded to categorise specific activities to undertake per colleague.

A Gantt chart is developed for visual reference as to what specific jobs are to be undertaken. Gantt charts are often presented like horizontal bar charts with each bar terminating between their respective start and end (deadline) dates – they still can overlap between duties. Some bars are lengthened to categorise a set of tasks to be done in the chart.

For example, 'Prototype Design Illustration' is when the colleague is to illustrate design ideas for the project's prototype.

The duty starts at the 17th of October and is due to be finished on the 27th of January.

See append document for 'Project Gantt Chart' for reference.

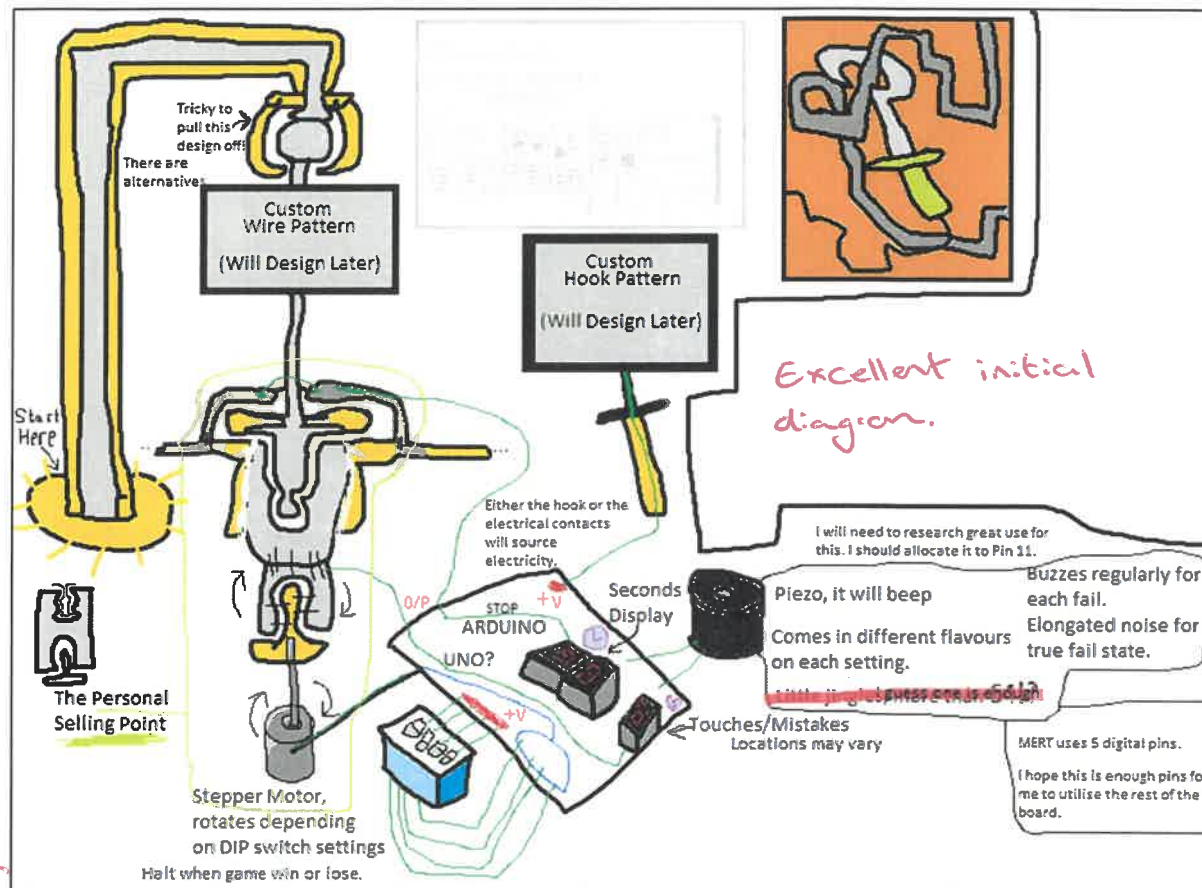
	Due Date
Research-to-Prototype Deadline	02/12/2022
Week Commencing Topic	17/10/2022
Research for Projects to Develop	17/10/2022
Gantt Chart Development	24/10/2022
Prototype-to-End Product Deadline	03/02/2023*

*Final Dates incl. End of Product Deadline (not shown) Pending

Project Plan

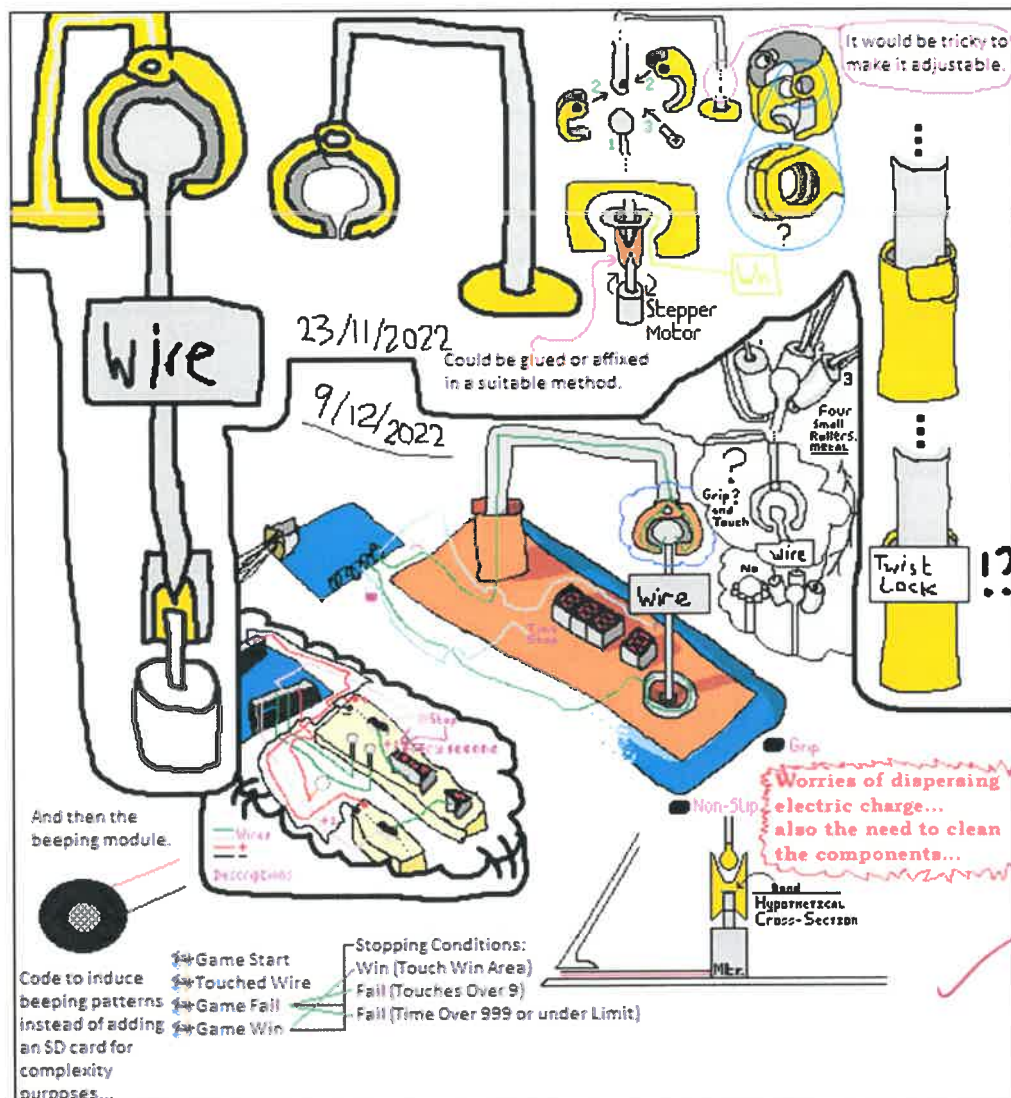
Not relevant. { The project plans can be reinterpreted initially as providing illustrations of prototypes of the projects in mind while describing specific aspects in general interest.

Initially, four different projects were taken, and one is then carefully decided for long-term development. See appendix 'List of Projects as Initial Decisions' for related information. *-Good.* The chosen project for development is the 'wire game', with a spinning twist with the wire.



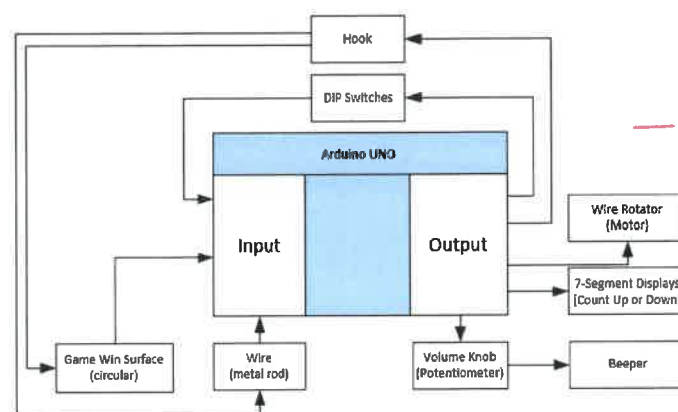
Although the project might contradict the requirement of being easy-to-medium difficulty and that being delicate to the project is not part of the task, conduction of electricity while displaying angular and manufactured components and a mistake counter can demonstrate respective abbreviations of STEM while the overall product should induce a form of interactive entertainment.

- Good justification.



The next image illustrates detailed ideas that aim closer to realistic ideas and what variations can be deemed possible to develop afterwards.

Block Diagram



The block diagram simplifies what elements of the project affect each other into boxes and subsequent arrows. Recipient aspects of blocks are attached to arrowheads, e.g: Output of Arduino UNO provides signals to the motor.

Research in the chosen EWB Challenge Design

Finding the origin of the traditional wire game is difficult to pinpoint. But the awareness to co-ordinate and correspond with 'Engineers Without Borders' (EWB) is to be considered during development of the project.

Engineers Without Borders is a global group of non-governmental organisations that helps participants develop and improve on engineering-based abilities and openly inspire development towards tackling global, physiological problems frequent across developing countries. Such examples include: extended pipe works for clean water; housing for families; photovoltaic installations; etc. But EWB is not limited to only supporting developing countries. (Engineers Without Borders, n.d.)

Eclectic use of commercially produced components such as crisp packets or plastic bottles can be proof that developing countries can make use of existing props – or recycling of such – to develop an entire project from scratch. And even crisp packets and aluminium foil can make use of electrical conduction. Such prophecies are mentioned since engineering makes use of said existing props and develop projects with them.

Project-Relevant Areas, Subjects and Technologies (Research)

Refer to appendix 'List of Projects as Initial Decisions' for project choices.

Calculation of Costs – including Bill of Materials (BoM)

Intended Qty.	Quantity	Colloquial Name	Cost of Indiv. Product	Product Code	Availability	Subtotal	Intended Subt.
4	4	7-Segment Display - Common Cathode	£ 4.100	588-639		£ 16.40	£ 16.40
50	5000	1kΩ Resistor	£ 0.007	739-7459		£ 35.00	£ 0.35
6	6	Large Breadboards	£ 5.950	239		£ 35.70	£ 35.70
1	1	Red Wire, 0.6mm - 100m	£ 9.480	CB19930		£ 9.48	£ 9.48
1	1	Green Wire, 0.6mm - 100m	£ 8.980	CB8R0252		£ 8.98	£ 8.98
1	1	Violet Wire, 0.6mm - 100m	£ 9.640	CB14053		£ 9.64	£ 9.64
1	1	Blue Wire, 0.6mm - 100m	£ 10.480	CB19927		£ 10.48	£ 10.48
1	1	Black Wire, 0.6mm - 100m	£ 9.180	CB19926		£ 9.18	£ 9.18
1	1	Orange Wire, 0.6mm - 100m	£ 10.680	CB19929		£ 10.68	£ 10.68
1	1	Yellow Wire, 0.6mm - 100m	£ 10.480	CB19932		£ 10.48	£ 10.48
1	1	Wire's Insulation Croppers - Cheap Option	£ 7.990	4786V	Equipment	£ 7.99	£ 7.99
1	1	Wire's Insulation Croppers - Standard Option	£ 14.990	5175V	Equipment	£ 14.99	£ 14.99
1	1	Arduino Uno - Rev3	£ 22.500	A000066	Personally Acquired	£ 22.50	£ 22.50
1	1	Hard Wire Bending Tool	£ -	n/a	Equipment	£ -	£ -
1	1	Hard Stainless Wire	£ 5.520	R600H00.600L30	Similar	£ 5.52	£ 5.52
4	25	CD4511BE IC - BCD to 7-Segment Decoder	£ 1.102	922-2903		£ 27.55	£ 4.41
5	25	74LS00 IC (Low-Schottky) - Quad 2-Input NAND Gates	£ 0.802	868-5884		£ 20.05	£ 4.01
5	25	74HC00 IC (High-Speed) - Quad 2-Input NAND Gates	£ 0.738	933-2748		£ 18.45	£ 3.69
5	25	74LS08 IC (Low-Schottky) - Quad 2-Input AND Gates	£ 0.859	923-2935		£ 21.48	£ 4.30
5	25	74HC08 IC (High-Speed) - Quad 2-Input AND Gates	£ 0.686	100-0760		£ 17.15	£ 3.43
5	25	74LS82 IC (Low-Schottky) - Quad 2-Input OR Gates	£ 0.750	923-2841		£ 18.75	£ 3.75
5	25	74HC82 IC (High-Speed) - Quad 2-Input OR Gates	£ 0.880	217-5645		£ 22.00	£ 4.40
4	4	74LS193 IC (Low-Schottky) - 4-Bit Up/Down Counter	£ 1.296	9006295		£ 5.18	£ 5.18
4	4	74HC193 IC (High-Speed) - 4-Bit Up/Down Counter	£ 0.992	9119614		£ 3.97	£ 3.97
3	100	Blinding Screws - 6mm	£ 0.216	BSN6	Unknown	£ 21.59	£ 0.65
4	700	Paperclips - 28mm	£ 0.010	n/a		£ 6.99	£ 0.04
6	100	Silver Tube Beads - Large	£ 2.430	LS222	No Stock	£ 243.00	£ 14.58
6	100	Silver Tube Beads - Small	£ 1.500	LS222	Similar	£ 150.00	£ 9.00
0	0	n/a	£ -	n/a		£ -	£ -
0	0	Note: I require one or more of the tube bead rollers as illustrated in the late 2022 design illustration.	£ -	n/a		£ -	£ -

"=SUM(J2:J31)"

"=J34*1.02"

"=ROUNDUP(J34/J31,0)-0.01"

Total £ 783.18 £ 283.77

Taxed Total (+2%) £ 798.84 £ 288.45

Tax to Nearest 99p £ 798.99 £ 288.99

The cost of individual products will be multiplied by quantities to output respective subtotals.

There are other components in the next list, but they are not given their prices.

For reference, an estimated total price will be given instead of standard research.

Generalised Names	Quantity
2-Pin Connector Sockets (can be polarised)	2
4-Pin Connector Sockets (can be polarised)	1
6-Pin Connector Sockets (can be polarised)	2
1k Resistors	18
14-Pin IC PCB-Friendly Holders	3
16-Pin IC PCB-Friendly Holders	1
4mm Phillips Button Head Screw	10
4mm Nut for the screws of above item	10
Steel Balls - 20mm	Any
Steel Balls - 30mm	Any
Steel Balls - 40mm	Any
Power Drill + some small drill bits with ability to indicate drill bit measurements	1
Metal Rod Bender	1
Small Saw to cut some of the items	1
Hacksaw (to cut metal rod)	1

Roughly the contents and frequency of equipment use would cost around £29 extra, leading to a grand 99p-based total of £268.99. *~ IS THIS PER UNIT?*

When perceiving mass production of the project, the price may drop to around £39.99 in consideration of component and equipment costs, but the price will still be rendered as too much when compared to other products of similar function. Such products (without the spinning wire) would cost around £9.49 as a commercial product.

~ How can you justify this cost?

Although the cost of the entire project ends up being expensive, the project must also be entertaining for the customer base.

~ I would like to see you calculate the unit cost.

References

Engineers Without Borders, n.d.. *Activities*. [Online]
Available at: <https://www.ewb-international.org/activities/>
[Accessed 31 May 2023].



University of Wales, Trinity Saint David

Gower College Swansea, Tycoch College

Designing an Affordable Means of Entertainment – Final Report

Ronan Pesarillo

PES15826963

Higher National Diploma (HND) in Electrical and Electronic Engineering

Monday, 12 June 2023

Group Project and Research Methods

Contents

Introduction.....	6
Research Reflection	6
Developmental Design Details	7
Friday, 18 November 2022.....	9
Wednesday, 23 November 2022	9
Physical Design Illustration Progress	9
Friday, 2 December 2022	9
Thursday, 8 December 2022	9
Friday, 9 December 2022	9
Wednesday, 14 December 2022.....	10
Tables for respective IC connections will be provided when implementing digits.....	10
Friday, 13 January 2023	11
Friday, 10 March 2023	12
New Life/Mistake Counter Iteration.....	12
An attempt to flash at 9, but flashes immediately on loop	13
Thursday, 16 March 2023	14
Code Fix to Successfully Require Expected Result with the 9-count Loop Condition....	14
Premature Counting Detected.....	15
Hook-Wire Contact Delay Outcome Partially Successful.....	16
Friday, 17 March 2023	18
Code Recap and Recollection 1	19
Current List of Implementations to Perform	20
Visual Prototype Recap 1	21
Wednesday, 22 March 2023.....	22
Sending Input has to be Digital for Digital Processing	22
Thursday, 23 March 2023	24
Pre-Game Warning Implementation.....	24
Friday, 24 March 2023	24
Wednesday, 29 March 2023.....	26
Basic Tones to Apply	26
Thursday, 30 March 2023	27
Visual Blanking	27
More Tones to Play and Grace Time Reduced.....	28
Win Condition Tone 1	29

Lose Condition Tone 1	29
Friday, 31 March 2023	30
Stepper Motor Imitation Research.....	30
Tuesday, 4 April 2023	32
Current Prototype View	32
Win and Lose Conditions, Minor Modifications.....	33
Wednesday, 3 May 2023	34
Renovating for the 74HC00.....	34
Proof of Texas Instruments Component Usage.....	35
List of Removed Details 1	35
Thursday, 4 May 2023	36
Getting to Fusion 360	36
Basic Steps to Schematic Design in Fusion 360.....	36
Friday, 5 May 2023	37
New Rod Design 1	37
How to Change Physical Material of Rod	37
Wednesday, 10 May 2023	38
How to Create a Wire Stand that should Rotate with the Step Motor.....	38
Friday, 12 May 2023	38
Uploading TinkerCAD Code to Arduino UNO.....	38
First Physical Prototype Test.....	39
Monday, 15 May 2023	45
Faults at Wiring and Code Modifications.....	45
Two Physical Prototype Tests	46
Wednesday, 17 May 2023	47
Breadboard 7-Segment Display Tester – Common Cathode.....	47
Breadboard Prototype Test Number 4	48
Breadboard Prototype Test Number 5	52
Breadboard Prototype Test Number 6	53
Breadboard Prototype Test Number 7 – Succeeded Draft Build.....	55
Breadboard Prototype Test Number 8	56
Thursday, 18 May 2023	57
Connector Pins on Circuit Schematic and Design.....	57
Pin Correspondence for Project.....	57
Breadboard Prototype Test Number 9	58

Breadboard Prototype Test Numbers 10 and 11	59
Metallic Contacts	64
Fusion 360 Current State of Circuit.....	65
Monday, 22 May 2023	66
Circuit Schematic of 22 May 2023, Iteration 1	66
Circuit Schematic of 22 May 2023, Iteration 2	67
Circuit Design of 22 May 2023, Iteration 1	67
Circuit Design of 22 May 2023, Iteration 2.....	68
Finding Comparability with Real Life Components	69
Hook Design, Iteration 3	70
Wire Stand Design, Log 2	70
Wire Stand Holders – Log 1	71
Tuesday, 23 May 2023	72
Circuit Design of 23 May 2023	72
Pushing the Circuit Design into 3D PCB	73
Wednesday, 24 May 2023	73
Temporary Circuit Board Extension from 2500 mil to 3000 mil	73
Inserting Components to a New Project	74
Beginning of Assembly and Physical Modifications	74
Assembly and Physical Modifications, Log 2	75
Thursday, 25 May 2023	76
Marble and Metallic Contact Visualisation	76
Making Do With A Household-Sized 3D Printer via. Jigsaw Cut Shaping.....	76
State of 3D Model.....	76
Preparation for 3D Printing	79
Friday, 26 May 2023	80
Before Mesh Printing.....	80
Exporting of PCB	80
Proof of 3D-Printed Model.....	80
Proof of Wire Top Holder (makeshift, not Fusion 360).....	81
Production of Wire Top Holder in Images	82
Makeshift Demonstration of Wire Game	83
Development Reflections	84
Voided Ideas	85
Final Design Details.....	86

Evaluation of Final Product Quality Assurance.....	86
Recommendations.....	87
Conclusions.....	87
References.....	88

Introduction

University of Wales, Trinity of Saint David once set a task in which to produce an easy-to-medium project for developing countries; corresponding with Engineers Without Borders (EWB) whilst providing an aspect of education by associating with 'Science, Technology, Engineering and Mathematics' (STEM) and entertainment within its purpose. Said project should be cost-efficient and simple to manufacture.

Four ideas were gathered for decision, and the buzz wire game was finally selected for year-long development. See append, document 'List of Projects as Initial Decisions' for corresponding ideas and information.

- Piano with capacitive touch interaction
- Breathalyser
- RPM counter measuring centrifugal speed via. Hall Effect
- Buzz wire game with basic appends

Review how we include appendices.

Research Reflection

When discussing about reuse of commercially produced products such as crisp packets or plastic bottles, what was meant from the documents was that the ability to produce projects from scrap material should elicit a mean of learning about component substitution regarding general project design.

- Relevance?

Developmental Design Details

Design of a **prototype wire game** started as an overall concept. was considered replicable and welcoming for modifications.

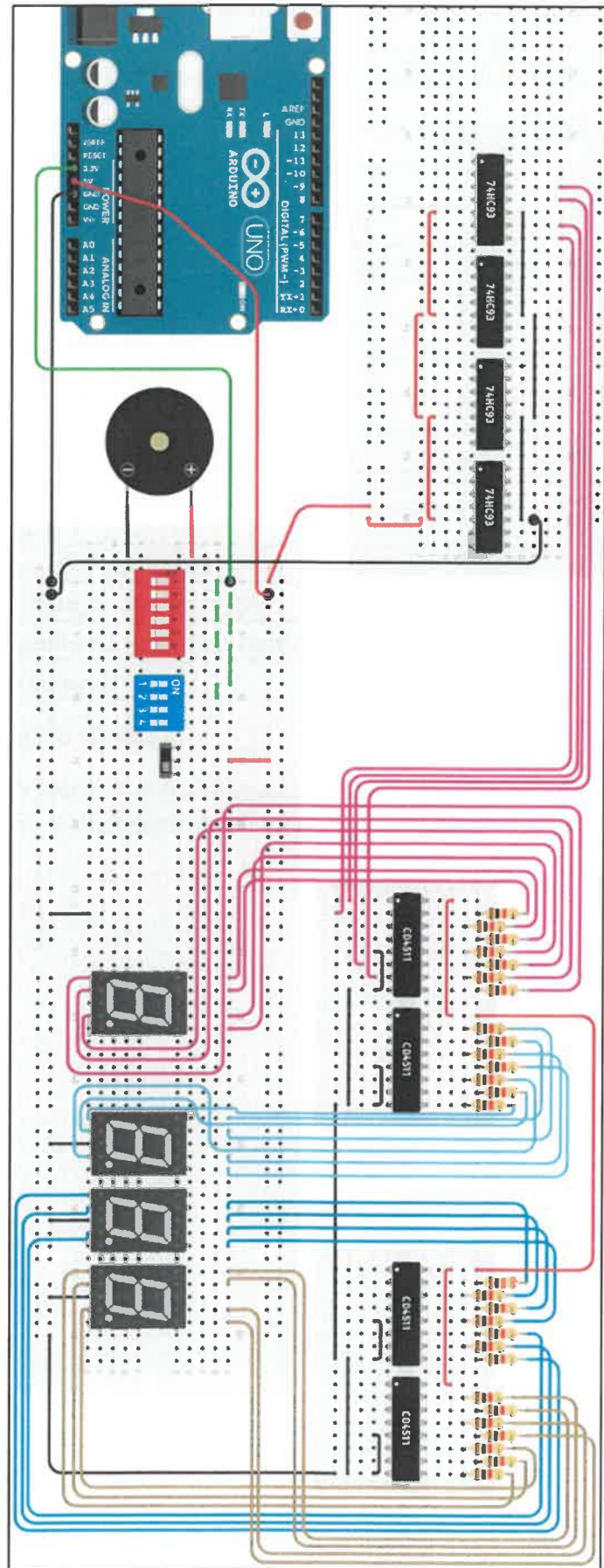
Visual design ideas were roughly illustrated using Microsoft Paint. Sketches include how components may be presented, captions, and an instruction page for adjustable settings.

TinkerCAD was used for early circuit design so to provide a convenient-enough visual simulation where the ideas begin to translate into another software. However, labour of component allocation to TinkerCAD proves oncoming tedium for the difficulty of the project.

The week's state of the prototype for the 'Wire Game' circuit board means to comprise of a life and timer system respectively (one 7-Segment display for the life system, and three 7-Segment displays for the timer system), including DIP switch settings for 8-bit gameplay customisation (not wired here) and a buzzer for output communication (also not wired).

Red wires indicate full voltage is sourcing to respective pins/rails, whereas black wires act as an entire ground rail. The magenta, blue(s), and brown rails mean to provide sourcing for the 7-Segment displays. The magenta wires connecting outside a 7-segment display means to act as a counter for how many lives are gone (wire contacts with hook) through the process of playing the game.

For context of components throughout the document, refer to Project (Appendix 2) for information.



However, with the mandatory obstruction of having to deal with external objectives such as other assignments against human capacity, not many of the illustrated ideas are to be implemented during development.

Most of the voided ideas will be due to busy schedules and time constraints.

Final deadlines per duty are now as follows:

Final Deadline Arrangements	Start Date	Due Date
Prototype Initiation Deadline	17/10/2022	22/12/2022
Research for Project Inspiration	17/10/2022	27/01/2023
Prototype Design Illustration	24/10/2022	27/01/2023
Gantt Chart Development	24/10/2022	09/06/2023
Bill of Materials	28/10/2022	03/02/2023
Block Diagram	11/11/2022	09/06/2023
Breadboarding	28/02/2023	09/06/2023
Schematic Design	03/04/2023	09/06/2023
C.A.D.	01/05/2023	09/06/2023
Prototype Testing	01/05/2023	09/06/2023
Adjustments to Builds	26/05/2023	09/06/2023
End Product Initiation Deadline	02/12/2022	09/06/2023
Flowchart	18/11/2022	03/02/2023

Refer to appendix document 'Gantt Chart for Project'.

Due to said reasons above, progress is noticeably decelerated.

It is also contributing to how most of the duties' deadlines are set to the final day of reporting.

↳ More reflection:

- What were the issues.
- How you over come them
- What would you do to avoid them in the future?

Friday, 18 November 2022

Continuation of progress report for November 11th. Initially, was suggested to peruse what project is to develop in favour of personal entertainment and prior knowledge before research and development is to occur for next week.

Refer to Project Appendix 4.1 for progress on Friday, 18 November 2022.

Wednesday, 23 November 2022

Refer to Project Appendix 4.2 for removed addition of three extra 7-segment displays.

Physical Design Illustration Progress

Further physical design choices were implemented in the sketch draft image regarding the wire holder for the wire game. It includes the decision to implement a flip lock or a twist lock for the wire holder, as seen in adjustable components on a tripod.

Refer to Initial Report, pages 4 and 5 for relevant illustrations.

Friday, 2 December 2022

Subheading for Wednesday, 23 November 2022 was modified to clarify about illustrations for prototype designs.

Updating explanations for components, mostly over 7400 ICs.

Added explanation of initials in ICs (Project Appendix 2).

Thursday, 8 December 2022

There may be the decision to replace the 74HC93 ICs with 74LS193 ICs.

Initially, confusion between LS ICs and HC ICs having differing input parameters obstructed intuition, only to notice and re-read that the HC ICs have the differing input parameters.

Benignly demonstrates confusion between IC types (see Project Appendix 2).

Friday, 9 December 2022

- Continuation of possible product designs, updated to include breadboard-based hypothesis.
- Initiation in developing 'Bill of Materials' (BoM) for breadboard prototype. Chosen product will include tax for each product selected where possible.
- Founded concern or weakness of the design idea that the product might cause increase of static discharge from the friction of the wire (the component the player is meant to guide across).
- According to John Vinod Khiatani (2022), the intended built product should follow (BS) EN 62115: 'Electric toys – Safety'.
- ❖ Interruptions with external products such as laser technology and magnetic interference may be conducted to test safety limits of the product.

— Good reference to industry standards. ✓

Wednesday, 14 December 2022

IC names are appended in titles for 'Initials in Integrated Circuits' to ensure clearer indication of journal contents.

Tables for respective IC connections will be provided when implementing digits.

Refer to Project Appendices 3.1.1 and 3.1.2 for tables used in context of Wednesday, 14 December 2022.

Continued development on other documents that act as explanatory convenience. Both of which indicate what modes will be activated depending on the DIP switch settings.

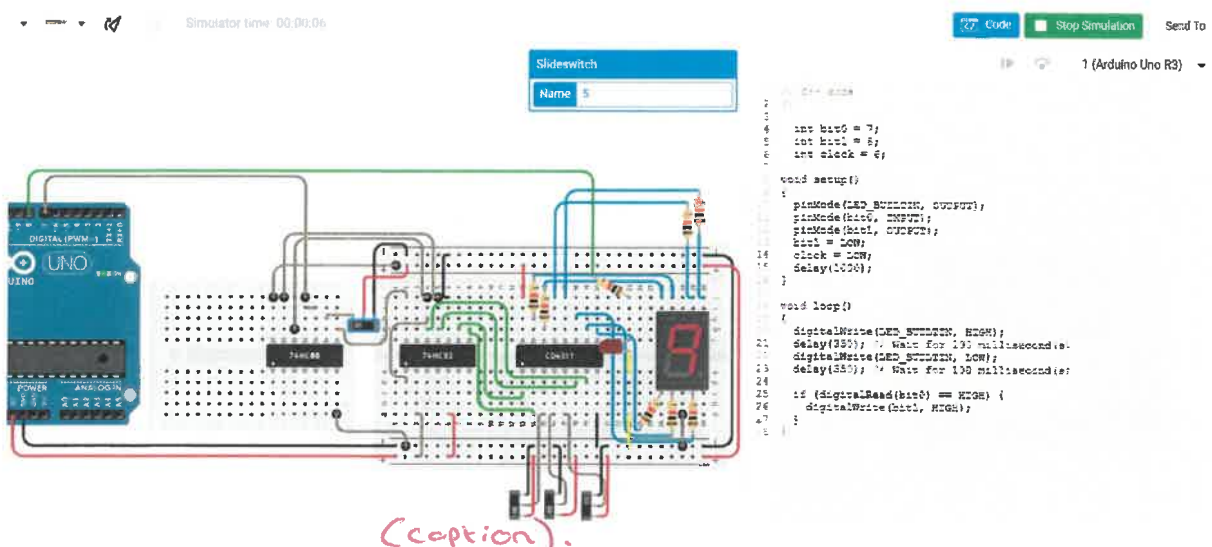
- Visio document that provides flowchart understanding for what logic and variables the project is meant to follow and display.
 - Unfortunately corrupted, resulting in missing decision and terminal symbols including their text.
- Excel spreadsheet used as truth table displays for different DIP settings.
 - Refer to Project Appendix 3.3 for displayed graph involved during development. ✓

Friday, 13 January 2023

- Updated Visio document to append the following:
 - Block Diagram that labels needed components for project and their status of input, output or mixed.
 - Fixed typographical error in 'Defining Bit Settings 2'.
 - Removed unnecessary shapes.
- Updated Gantt chart to add more tasks' initiations and deadlines.
- Duplicated TinkerCAD single counter to test life/mistake counter logic.

Research at the time finds new types of screws named 'book screws' and 'binding screws'.

While they are metallic, it is of utmost faith that direct electrical conduction occurs from the hook to the wire, the stand and back to the Arduino UNO. On the other hand, a comparator circuit may be installed, considering the chances of high-frequency input interference.



During the course of duplicating another counter test for a different purpose, configuration to set the other counter to count up to '9' was partially successful.

Previously was using three pairs of NAND gates, but the current displayed wiring configuration took two pairs of NAND gates to function.

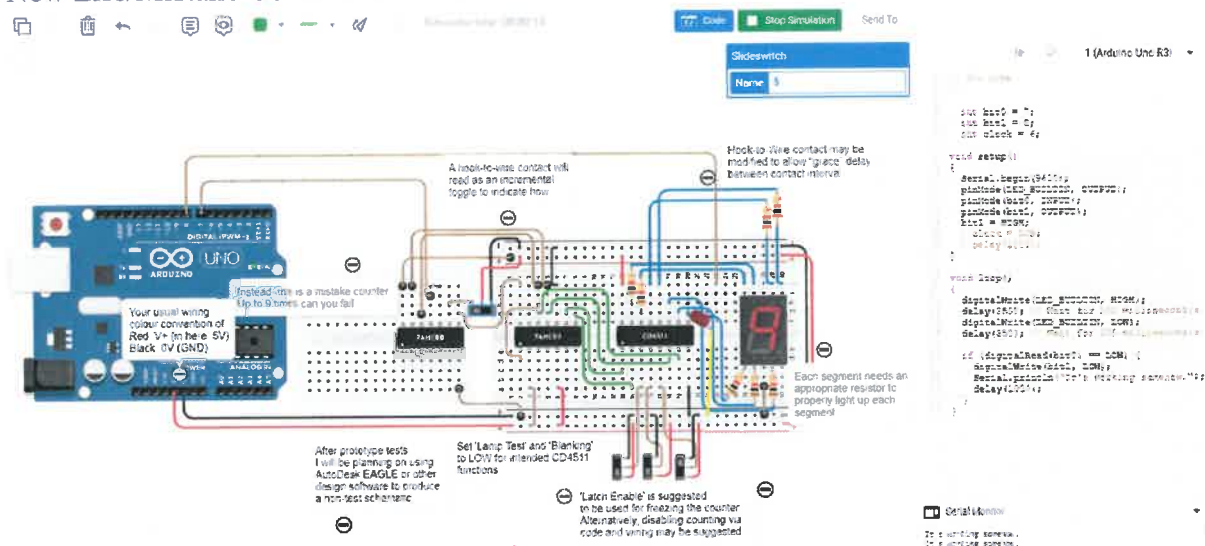
Several refreshes from reading code from past designs remind to use digitalRead() and digitalWrite() functions instead of typing only variables, operands, and brackets.

-Excellent log of progress.
+ Reflection.

Friday, 10 March 2023

- ☀ Developing hypothetical scenario of conditions, variables and code on a new notepad file.
- + Fixed typo on the title of Friday, 13 January 2023.
- + Continuation on Visio development.
- + Continuation with prototype modification.
 - Visualising block logic for what happens when the hook touches the wire.
 - More logic gates may have to be implemented for stricter logic specifications.
- Gantt Chart amendments regarding lack of progress on Project.
 - Extending 'Block Diagram' deadline.
- Decision over feature cuts to compromise favoured on time constraints.
 - Timer might have to be removed due to initial demand for excess components and extra complication of code.

New Life/Mistake Counter Iteration

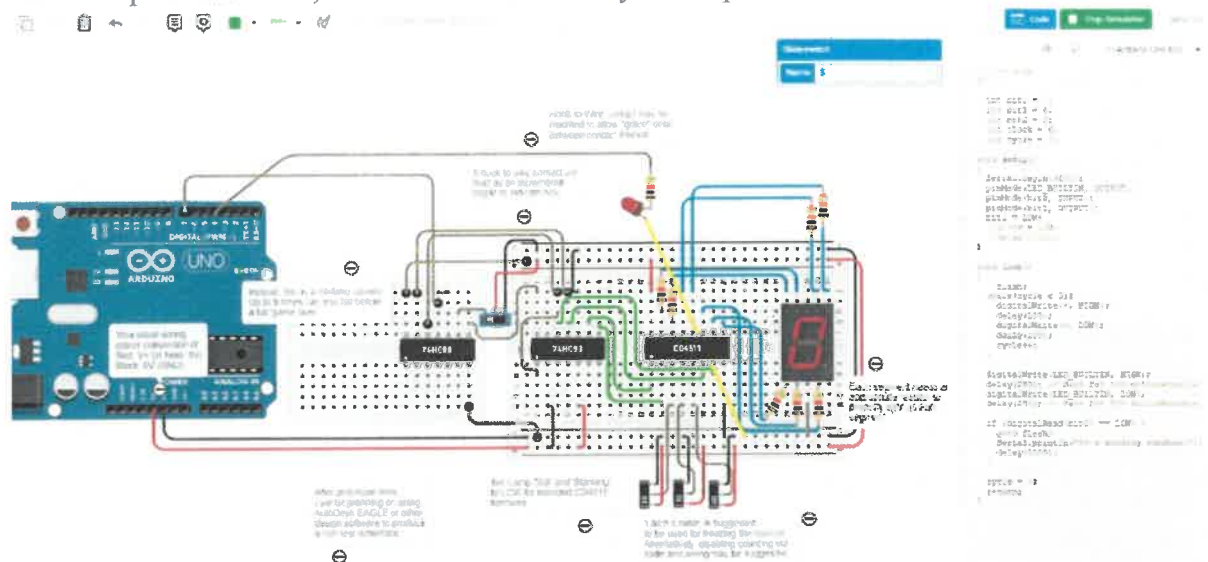


While still in the process of prototype development on manual counter incrementation, sending a signal to one of an Arduino UNO's digital pins does activate conditions for when '9' is read in the counter.

A temporary implementation requires that the red LED next to the counter must be lit up after the counter maxes out.

A solution found from it is that in line 26 of the provided coded, swapping out the name of the variable to its assigned number will enable the lighting of the LED.

An attempt to flash at 9, but flashes immediately on loop

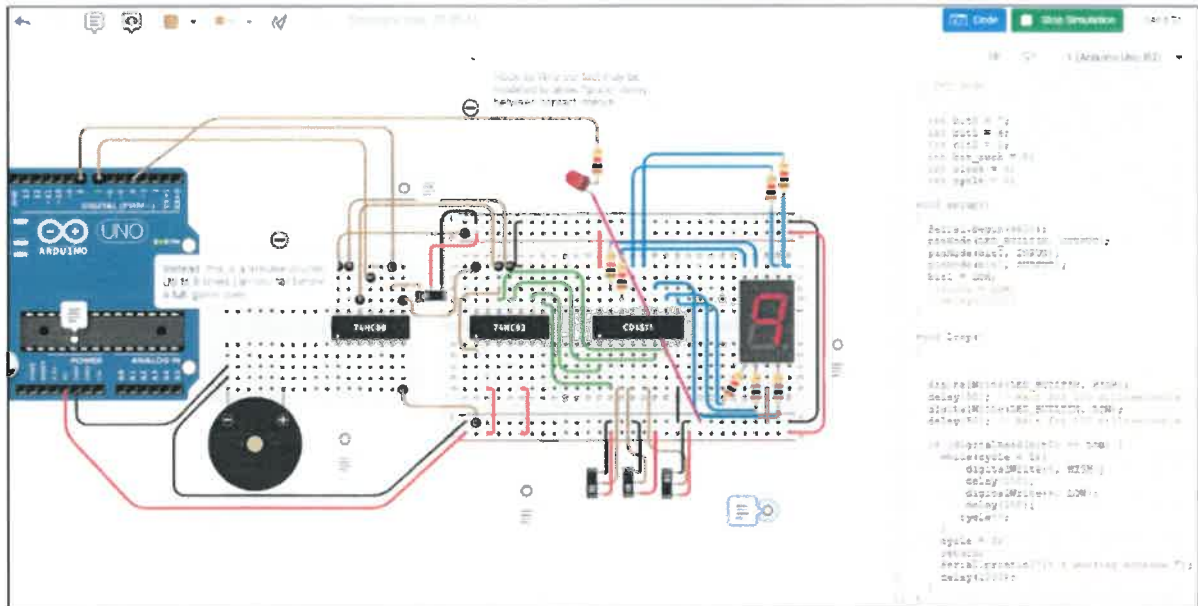


Brief research suggests that **goto** results in the risk of code reading in jumbled order. ✓

Thursday, 16 March 2023

- Continued development on TinkerCAD and C++ code.
- Incrementation up to 9 to trigger an alternate event.

Code Fix to Successfully Require Expected Result with the 9-count Loop Condition

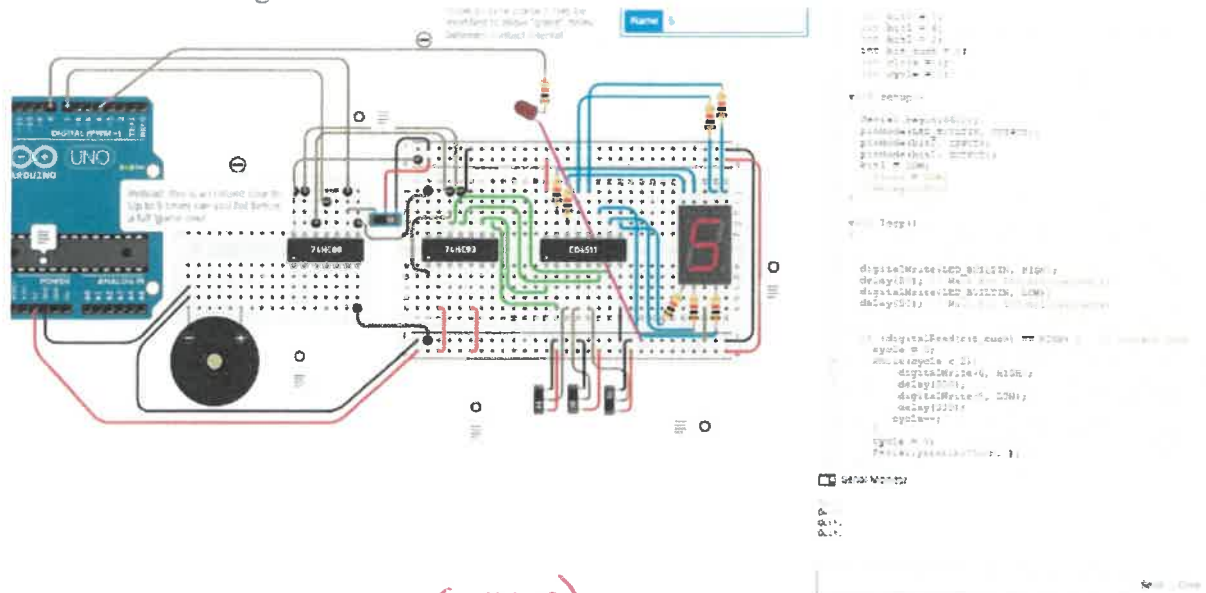


The attempt of using `goto` (as seen in previous page) was later removed, so a nested if-while statement takes place instead. What is not shown on the lower-right of the image is that the Serial Monitor prints a line every 1 second interval from line 41.

Additional conditions inside if-statements are to be implemented so to avoid hypothetical concerns with code lines simultaneously executing. *- Good.*

A piezo is prematurely placed on the smaller breadboard for future sound purposes.

Premature Counting Detected



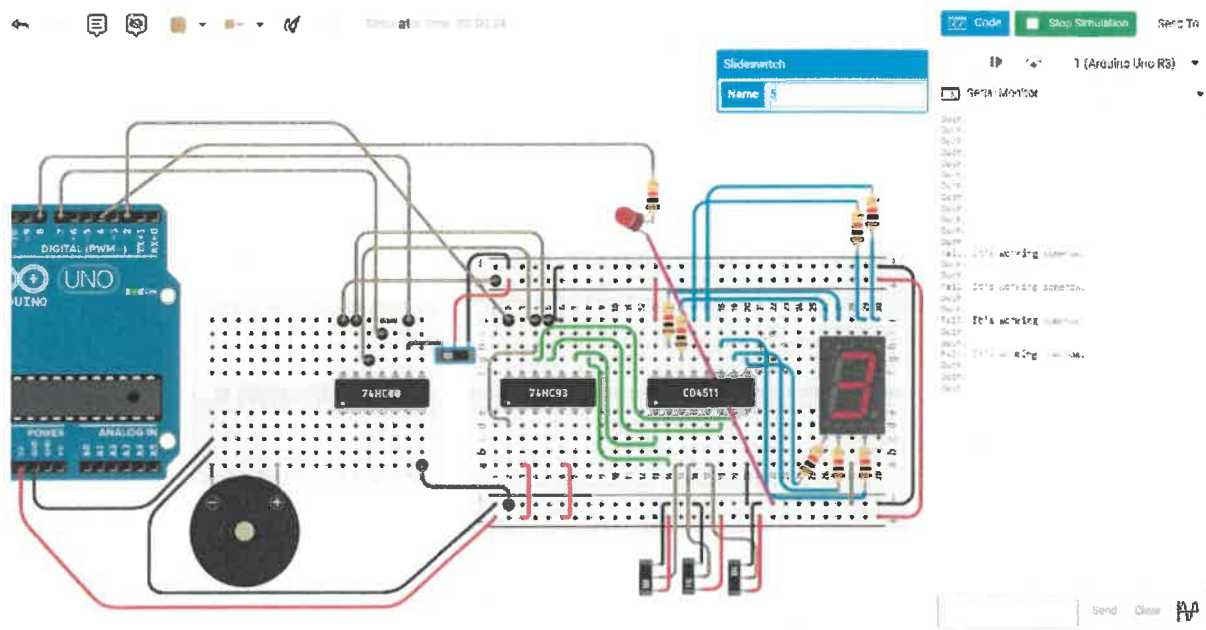
(caption)

Adding a new paragraph of code in which hook-wire contact occurs, physically a wire from pin 9 of the 74xx00 is added. The condition of 9-counting also regards the condition of contact.

However, when administering 'hook-wire contact' stimuli when the processes of grace happen, the counter increments regardless of other events happening. This results in the Serial Monitor emitting its requested string of characters from line 42 less than the number the counter shows.

Lines 32 to 43 are due to be shifted below the lines that regard when the counter reaches 9, but will be implemented later.

But when the stimuli administering is continuous as opposed to manually controlled, the counter ends up counting past 15 (2^4 bits minus 1) in which the counter's number does not equal the amount of 'Ouch.' lines emitted from the Serial Monitor.



The code for which intended maximum capacity reaches does not cause a hard lock against further functions.

Friday, 17 March 2023

- TinkerCAD and C++ continuation.
 - Implemented while loop to prevent further logic supplementation.
- Fixed grammar on Thursday, 16 March 2023 update bullet point.

The `lock_event` integer variable (line 10) is implemented for indicative convenience, noting when the program locks into a loop with no events being able to control the outcomes of the program.

Once the program reads that the number 9 shows on the display counter after grace time, the next hook-wire contact event will invoke a different event where the game tells the player that they lost. Preferably, a piezo or beeper will indicate said event differently from normal hook-wire contact and (later on to implement) pre-game conduct.

Outside TinkerCAD, a breadboard is attached with stripped wires and are checked for continuity between ends.

See next two pages to recap on what functions perform what action in C++ and for what group of lines perform what actions.

Right – current program code of project.

```

1 // -- Code
2
3 int x1Pin = 7;
4 int x2Pin = 4;
5 int x3Pin = 5;
6 int attack = 0;
7 int clock = 0;
8 int cycle = 0;
9 int lock_event = 0;
10
11 void setup()
12 {
13   Serial.begin(9600);
14   pinMode(LED_BUILTIN, OUTPUT);
15   pinMode(x1Pin, INPUT);
16   pinMode(x2Pin, OUTPUT);
17   pinMode(x3Pin, INPUT);
18   pinMode(attack, OUTPUT);
19   digitalWrite(LED_BUILTIN, LOW);
20   digitalWrite(x2Pin, LOW);
21   digitalWrite(attack, LOW);
22   digitalWrite(x3Pin, LOW);
23 }
24
25 void loop()
26 {
27   digitalWrite(LED_BUILTIN, HIGH);
28   delay(500); // Wait for LED to turn on (500ms)
29   digitalWrite(LED_BUILTIN, LOW);
30   delay(500); // Wait for LED to turn off (500ms)
31
32   if (digitalRead(x1Pin) == HIGH && digitalRead(x2Pin) == LOW) {
33     cycle = 0;
34     lock_event = 0;
35     while(cycle < 5) {
36       digitalWrite(x2Pin, HIGH);
37       delay(100);
38       digitalWrite(x2Pin, LOW);
39       delay(100);
40       cycle++;
41     }
42
43     cycle = 0;
44     while(lock_event == 1) {
45       Serial.println("Locking sequence!");
46       delay(100);
47     }
48   }
49
50   if (digitalRead(x3Pin) == HIGH) {
51     Serial.println("Lost!");
52     digitalWrite(x2Pin, HIGH);
53     cycle = 0;
54     digitalWrite(x2Pin, LOW);
55     while(cycle < 5) {
56       digitalWrite(x2Pin, HIGH);
57       delay(100);
58       digitalWrite(x2Pin, LOW);
59       delay(100);
60       cycle++;
61     }
62
63     cycle = 0;
64     Serial.println("Lost!");
65   }
66 }

```

Code Recap and Recollection 1

Characters with double forward-slashes before them will render code as comments. See lines, 1, 2, 22, 23, 29, 31, 34 and 51 as functioning examples. `//`

Lines 4 to 10 signify the type of variable, and is then named after signifiers have been typed.

`int` renders the variables as integers i.e: 0, 1, 2, 3...

`const` ensures that the variable always contain an assigned value.

`=` is to modify the variable to contain the right-hand side case.

`==` compares the left-hand side variable with a specified case.

`void setup()` – lines of code that will occur once before the program reads lines under `void loop()`.

`void loop()` – lines of code that will occur repeatedly inside its braces/curly brackets.

`Serial.begin([speed parameter number])` – enables the functions of Serial Monitor to read and write messages, serial-based I/O, etc. Often the number '9600' is typed in as the baud rate of digital communication and to let the Arduino's downloading device (such as a PC or a laptop with appropriate software installed) or other same-number Arduino devices communicate stably. (Programming Electronics Academy, 2021)

`Serial.println("[text]")` – makes the Serial Monitor output text as written in the brackets under a new line as opposed to appending with other outputted characters. Because 'ln' is appended in this line of code, Carriage Return (`\r`) and New Line (`\n`) appends afterwards. They function as the character selector

`pinMode([variable], INPUT)` – assigns the variable set in the list of digital pins on an Arduino board as an input.

`pinMode([variable], OUTPUT)` – same as above, but as an output.

You can set `pinMode`-assigned variables as HIGH or LOW than just an integer, which might assign them as Boolean.

`digitalWrite([variable], HIGH)` – writes its specified (`pinMode`-assigned) variable to HIGH (or in Boolean, '1').

`digitalWrite([variable], LOW)` – same as above, but to LOW (or in Boolean, '0').

`delay([number])` – sets a delay on program's processing in milliseconds before other lines of code get executed.

`if([condition])` – sets a condition that, if met, makes the program read the line's brace/curly bracket-enclosed contents once.

`While([condition])` – almost the same as above, but is often accompanied with an incremental or decremental line of code like `[variable]++` in its brace/curly bracket-enclosed lines for repeating a sequence of code all the way to its end until the condition is not met.

– Solid level of detail.
– Although formatting could be improved.

See Page 21 for context of the game's code.

Lines 28 to 31 might have to be implemented for pre-game and post-game conditions.

Lines 51 to 64 states that when the hook touches the wire, a sequence of events occur while allowing the player to continue through the game without taking more than one increment of the counter. Grace timing prevents excess incrementing.

Said sequence of events is that some output source(s) will emit to indicate contact e.g: counter will increment one, then a buzzer will play a sequence of beeps.

Lines 34 to 49 performs similarly to above line section, but plays a different tune and prevents further stimuli from changing the game.

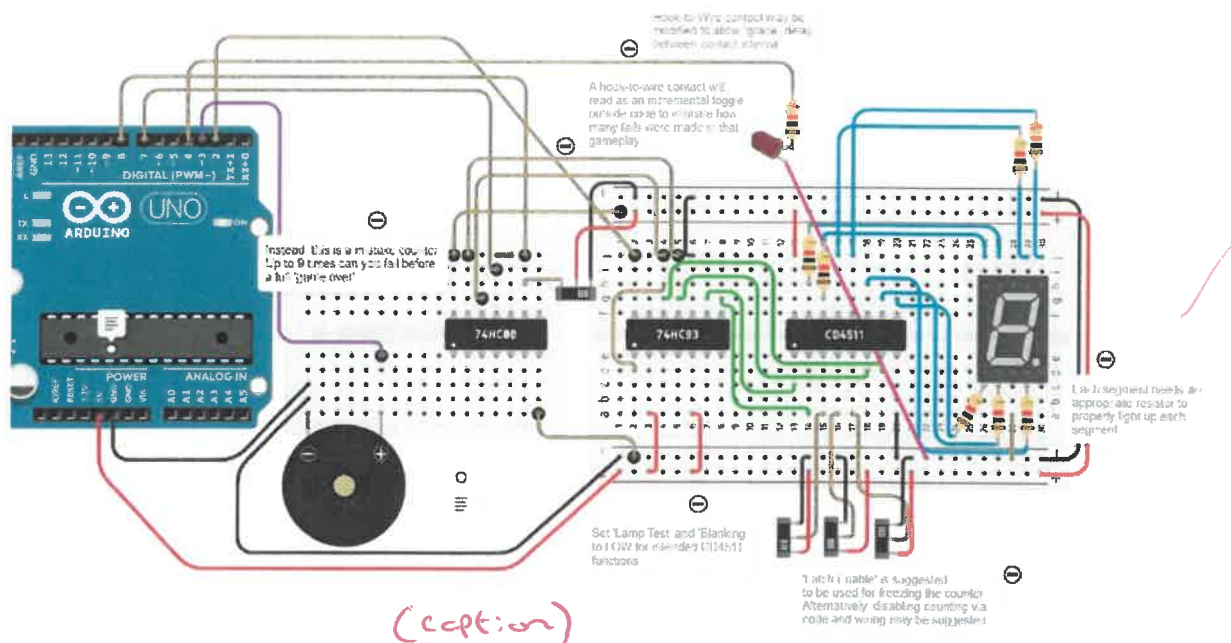
Current List of Implementations to Perform

- For duplication of lines 34 to 49, a win condition is to be implemented and play a different sequence of beeps. (requires beeper component)
- 'If' statements will include more conditions to further specify where contact with either wire or winning surface happens.
- Within 'void setup()', prevention of contact during start-up will cause beeps to tell the player to let contact be removed from the hook and place on a starting point.
- A section of code lines that perform a "ready, get set, go" sequence may not be applicable unless there exist code combinations that allow jumping back from a point of code whenever any point of the duration happens. ✓

See next page for recap's state of TinkerCAD prototype.

- These could all be summarized.

Visual Prototype Recap 1



Red and black wires signify as power (5V) and ground/common (GND) wires respectively.

The green wires connected from 74xx93 to CD4511 act as communication to specify in binary what number will show up, built to know how to increment via. powering of clock pins (1 and 14).

The blue wires on the right send a regulated amount of current per segment of the 7-segment display and lights up segments depending on requested binary-to-decimal value by the CD4511. 1k resistors (brown, black, red and gold stripes) reduce current to prevent damage to further components.

The three switches on the bottom-right are 'Lamp Test', 'Blanking', and 'Latch Enable'.

'Lamp Test' (pin 3) lights up all segments of a 7-segment display when set to 'LOW'.

'Blanking' (pin 4) disables lighting of said segments when set to 'LOW'.

'Latch Enable' (pin 5) prevents the display from changing until set to 'HIGH'.

Brown wires have pending designations.

Mostly are used as varying Boolean signals or for board-to-board connections.

A purple wire is for the beeper, in which development is to occur later.

The pink wire on the right connected to pin 4 is currently in an improperly aligned point intentionally for visual indication of hook-wire contact.

Wednesday, 22 March 2023

- TinkerCAD and C++ continuation.
- Modifying end-game conditions in code.

```
if (digitalRead(bit_couch) == HIGH && digitalRead(bit0) == LOW) { // Contact Game Over
  Serial.println("Game Set B");
  cycle = 0;
  lock_event = 1;
  while(cycle < 8){
    digitalWrite(4, HIGH);
    delay(100);
    digitalWrite(4, LOW);
    delay(100);
    cycle++;
  }
  cycle = 0;
  while(lock_event == 1){
    Serial.println("Fail. It's working somehow.");
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(1000);
  }
}
```

The above paragraph of code is to be duplicated so to create the next game-end condition. The change made to the paragraph is that the condition of taking contact with the win surface is removed regarding a hypothesis that after the code of the if-statement, the if-statement followed afterwards during the exiting of the current condition will be carried out.

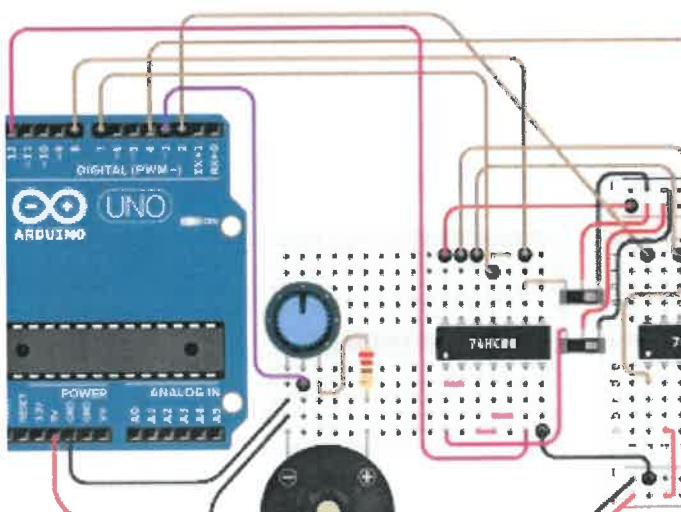
The duplication should be edited so that the game tells the player that they have “won” the game instead of giving them fail conditions that will not let them win the game.

```
if(digitalRead(8) == 0 || digitalRead(12) == 1){
  while(digitalRead(8) == 0 || digitalRead(12) == 1){
    digitalWrite(LED_BUILTIN, HIGH);
    delay(300);
    digitalWrite(LED_BUILTIN, LOW);
    delay(300);
  }
}
```

The code on the left is to be removed so to reduce debugging complications. Currently is part of the conflicting issue as declared from below modification.

Sending Input has to be Digital for Digital Processing

After deeper examination of the prototype, another hypothesis in discovery is that if a digital input pin does not read a 74xx00-smoothened output, then the Arduino breadboard will be unable to determine whether the “data given” is HIGH (1) or LOW (0).



The pink wires as linked (not the purple wire connecting to the potentiometer) is linked as displayed so that digital pin 12 will initiate events based on the if-statement regarding in priority to that pin at a HIGH condition.

For result of connection, see next page. ✓

Three tests have been carried out in order: — *Excellent.*

- Hook-wire contacts then win-surface contact individually;
- Hook-wire contact and win-surface contact in respective switch-ON order;
- Win-surface contact only.

All three tests have performed their individual events successfully without providing excess outcomes.

```

26 void loop()
27 {
28
29   if (digitalRead(win_hit) == HIGH) { // Contact WIN
30     Serial.println("Game Set A");
31     cycle = 0;
32     lock_event = 1;
33     while(cycle < 5){
34       digitalWrite(4, HIGH);
35       delay(100);
36       digitalWrite(4, LOW);
37       delay(100);
38       cycle++;
39     }
40     cycle = 0;
41     while(lock_event == 1){
42       Serial.println("Win. It's working somehow.");
43       digitalWrite(LED_BUILTIN, HIGH);
44       delay(1000);
45       digitalWrite(LED_BUILTIN, LOW);
46       delay(1000);
47     }
48   }
49
50   if (digitalRead(bit_couch) == HIGH && digitalRead(bit0) == LOW) { // Contact Game Over
51     Serial.println("Game Set B");
52     cycle = 0;
53     lock_event = 1;
54     while(cycle < 5){
55       digitalWrite(4, HIGH);
56       delay(100);
57       digitalWrite(4, LOW);
58       delay(100);
59       cycle++;
60     }
61     cycle = 0;
62     while(lock_event == 1){
63       Serial.println("Fail. It's working somehow.");
64       digitalWrite(LED_BUILTIN, HIGH);
65       delay(1000);
66       digitalWrite(LED_BUILTIN, LOW);
67       delay(1000);
68     }
69   }
70
71   if (digitalRead(bit_couch) == HIGH && digitalRead(bit0) == HIGH) { // Contact Couch
72     Serial.println("2AP");
73     digitalWrite(2, HIGH);
74     cycle = 0;
75     digitalWrite(2, LOW);
76     while(cycle < 2){
77       digitalWrite(4, HIGH);
78       delay(300);
79       digitalWrite(4, LOW);
80       delay(300);
81       cycle++;
82     }
83     cycle = 0;
84     Serial.println("Couch.");
85   }
86 }

```

The order of how the if-statement events carry out appear to prioritise over the win condition since the negative if-statements are set in a quantified order and that the win condition comes after them.

A reflection towards the if-statement conditions could do about the hook's contact to the win surface. But considering a general nature of hook-wire games, the conditions can be set as is programmed.

For consideration of finalising code, the Serial lines may be removed to shorten memory usage.

✓ — *Good analysis.*

Thursday, 23 March 2023

- TinkerCAD and C++ continuation.
- Updated 'Code Recap' to include `Serial.println`.
- Implementing pre-game warning before gameplay.

Pre-Game Warning Implementation

At the start-up of the game when either two contacts are currently touching the hook, the game warns the player to remove the hook from either wire or win surface.

The `void setup()` code displayed in the picture below will be added again below the `pinMode` assignments.

As the implementation is successful, the current state of code shows that only the built-in LED of the Arduino flashes for 600 millisecond cycles until both contact events are set to LOW.

Error in the code displays that the `bit_ouch` variable is involved twice instead of once and regarding the `win_hit` variable.

```
void setup()
{
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(bit_0, INPUT);
  pinMode(bit_ouch, INPUT);
  pinMode(win_hit, INPUT);

  pinMode(indctr, OUTPUT);
  pinMode(attack, OUTPUT);

  if(digitalRead(bit_ouch) == HIGH || digitalRead(bit_ouch) == HIGH){
    while(digitalRead(bit_ouch) == HIGH || digitalRead(bit_ouch) == HIGH){
      digitalWrite(LED_BUILTIN, HIGH);
      delay(300);
      digitalWrite(LED_BUILTIN, LOW);
      delay(300);
    }
  }
}
```

Friday, 24 March 2023

- TinkerCAD and C++ continuation.
- Updated Thursday, 16 March 2023 contents. Changed first sub-heading title for specification purposes.
- Updated Friday, 17 March 2023; 'Visual Prototype Recap 1' for purpose of the pin 4 LED with the pink wire on the prototype.
- Fixed bug regarding sub-heading subject from Thursday, 23 March.
- Redundant modifications to `digitalWrite()` arguments applied next page.

The code displayed on the right is the current `void setup()` code before piezo/buzzer-related lines are implemented.

The next change to the code involves `digitalWrite()` arguments. Image next page.

```
void setup()
{
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(bit_0, INPUT);
  pinMode(bit_ouch, INPUT);
  pinMode(win_hit, INPUT);

  pinMode(indctr, OUTPUT);
  pinMode(attack, OUTPUT);

  while(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
    if(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
      while(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
        digitalWrite(LED_BUILTIN, HIGH);
        delay(300);
        digitalWrite(LED_BUILTIN, LOW);
        delay(300);
      }
    }
    // Beeping code (to do later)
    delay(1000); // For now. Simple delay, then hopefully implementing buzzer code.
  }
}
```

```

14 int bit0 = 7;           // Condition that '9' shows.
15 int bit_ouch = 8;      // Hook-wire contact occurs.
16 int win_hit = 12;      // Hook touches winning surface.
17
18 int indctr = 4;        // Output visual indicator for hook-wire contact (temporary).
19 int attack = 2;        // Makes the counter increment by one and indicates hook-wire contact.
45 void loop()
46 {
47
48     if (digitalRead(win_hit) == HIGH) { // Contact WIN
49         Serial.println("Game Set A");
50         cycle = 0;
51         lock_event = 1;
52         while(cycle < 5){
53             digitalWrite(indctr, HIGH);
54             delay(100);
55             digitalWrite(indctr, LOW);
56             delay(100);
57             cycle++;
58         }
59         cycle = 0;
60         while(lock_event == 1){
61             Serial.println("Win. It's working somehow.");
62             digitalWrite(LED_BUILTIN, HIGH);
63             delay(1000);
64             digitalWrite(LED_BUILTIN, LOW);
65             delay(1000);
66         }
67     }
68
69     if (digitalRead(bit_ouch) == HIGH && digitalRead(bit0) == LOW) { // Contact Game Over
70         Serial.println("Game Set B");
71         cycle = 0;
72         lock_event = 1;
73         while(cycle < 5){
74             digitalWrite(indctr, HIGH);
75             delay(100);
76             digitalWrite(indctr, LOW);
77             delay(100);
78             cycle++;
79         }
80         cycle = 0;
81         while(lock_event == 1){
82             Serial.println("Fail. It's working somehow.");
83             digitalWrite(LED_BUILTIN, HIGH);
84             delay(1000);
85             digitalWrite(LED_BUILTIN, LOW);
86             delay(1000);
87         }
88     }
89
90     if (digitalRead(bit_ouch) == HIGH && digitalRead(bit0) == HIGH) { // Contact Ouch
91         Serial.println("ZAP");
92         digitalWrite(attack, HIGH);
93         cycle = 0;
94         digitalWrite(attack, LOW);
95         while(cycle < 2){
96             digitalWrite(indctr, HIGH);
97             delay(300);
98             digitalWrite(indctr, LOW);
99             delay(300);
100             cycle++;
101         }
102         cycle = 0;
103         Serial.println("Ouch.");
104     }
105 }

```

While optional to modify in context, the `digitalWrite()`'s first variables that were formerly numbers are assigned as respective variable names.

Wednesday, 29 March 2023

- TinkerCAD and C++ continuation.
- Implementation of `tone()` function on `void setup()`.

Basic Tones to Apply

```

4:  int bit0 = 7;           // Condition that 'S' shows.
5:  int bit_ouch = 8;       // Hook-wire contact occurs.
6:  int win_hit = 12;       // Hook touches winning surface.

7:
8:
9:  int indctr = 4;         // Output visual indicator for hook-wire contact (temporary).
10: int attack = 2;         // Makes the counter increment by one and indicates hook-wire contact.
11: int see_count = 13;     // Determines if the player is allowed to view the counter's number.
12:
13:
14: int cycle = 0;          // Appears when repetition is needed.
15: int lock_event = 0;     // Hard Lock for End Conditions|

16:
17: void setup()
18: {
19:   Serial.begin(9600);
20:   pinMode(LED_BUILTIN, OUTPUT);
21:   pinMode(bit0, INPUT);
22:   pinMode(bit_ouch, INPUT);
23:   pinMode(win_hit, INPUT);
24:
25:   pinMode(indctr, OUTPUT);
26:   pinMode(attack, OUTPUT);
27:
28:   while(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
29:     if(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
30:       while(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
31:         digitalWrite(LED_BUILTIN, HIGH);
32:         tone(3, 330);
33:         delay(400);
34:         digitalWrite(LED_BUILTIN, LOW);
35:         noTone(3);
36:         delay(400);
37:       }
38:       tone(3, 100);
39:       delay(1000);
40:     }
41:   }
42:
43:   tone(3, 750);
44:   delay(100);
45:   noTone(3);
46: }

```

Of the code from `void setup()`, lines of `tone(pin, frequency)` which play a square wave of 50% duty with select parameters and `noTone(pin)` which stops occurring tones, are implemented and easily function as intended.

Lines 28 to 36 indicate that immediate contact with either wire or win surface from the hook is detected before the game begins, thus demand to remove will occur.

Once those requirements are met (including the completion of the beeping cycle), a droning beep of 100Hz plays for one second in case mentioned contact(s) manually persist. After adhering to protocol, the game begins by beeping 750Hz for 100 milliseconds. ✓

Thursday, 30 March 2023

- TinkerCAD and C++ continuation.
- Modifying Blanking of 7-Segment Display

Visual Blanking

A yellow wire connected from digital pin 13 of the Arduino to pin 4 of the CD4511 replaces the temporarily implemented toggle switch method.

The variable labelled `see_count` associates with whether the 7-segment display will light up in accordance with the CD4511's logic of blanking the display.

```

1  int bit0 = 7; // Condition that 'S' shows.
2  int bit_push = 8; // Hook-wire contact occurs.
3  int win_bit = 12; // Hook touches winning surface.
4
5
10 int indout = 4; // Output visual indicator for hook-wire contact (temporary).
11 int attack = 3; // Makes the counter increment by one and indicates hook-wire contact.
12 int see_count = 10; // Determines if the player is allowed to view the counter's number
13
14
15 int cycle = 0; // Appears when repetition is needed.
16 int lock_event = 1; // Hard Lock for End Conditions
17
18 void setup()
19 {
20   Serial.begin(9600);
21   pinMode(LED_BUILTIN, OUTPUT);
22   pinMode(bit0, INPUT);
23   pinMode(bit_push, INPUT);
24   pinMode(win_bit, INPUT);
25
26   pinMode(indout, OUTPUT);
27   pinMode(attack, OUTPUT);
28   pinMode(see_count, OUTPUT);
29
30   digitalWrite(see_count, LOW);
31
32   while(digitalRead(bit_push) == HIGH || digitalRead(win_bit) == HIGH){
33     if(digitalRead(bit_push) == HIGH || digitalRead(win_bit) == HIGH){
34       digitalWrite(LED_BUILTIN, HIGH);
35       tone(3, 850);
36       delay(400);
37       digitalWrite(LED_BUILTIN, LOW);
38       noTone(3);
39       delay(400);
40     }
41     tone(3, 100);
42     delay(1000);
43   }
44   digitalWrite(see_count, HIGH);
45   tone(3, 750);
46   delay(100);
47   noTone(3);
48 }
49
50 void loop()
51 {
52   if (digitalRead(win_bit) == HIGH) { // Contact WIN
53     Serial.println("Game Set A");
54     cycle = 0;
55     lock_event = 1;
56     while(cycle < 5){
57       digitalWrite(indout, HIGH);
58       delay(100);
59       digitalWrite(indout, LOW);
60       delay(100);
61       cycle++;
62     }
63     cycle = 0;
64     while(lock_event == 1){
65       Serial.println("Win. It's working somehow.");
66       digitalWrite(LED_BUILTIN, HIGH);
67       delay(1000);
68       digitalWrite(LED_BUILTIN, LOW);
69       delay(1000);
70     }
71   }
72
73   if (digitalRead(bit_push) == HIGH || digitalRead(bit0) == LOW) { // Contact Game Over
74     Serial.println("Game Set B");
75     digitalWrite(see_count, LOW);
76     cycle = 0;
77     lock_event = 1;
78     while(cycle < 5){
79       digitalWrite(indout, HIGH);
80       delay(100);
81       digitalWrite(indout, LOW);
82       delay(100);
83       cycle++;
84     }
85     cycle = 0;
86     while(lock_event == 1){
87       Serial.println("Fail. It's working somehow.");
88       digitalWrite(see_count, LOW);
89       digitalWrite(LED_BUILTIN, HIGH);
90       delay(1000);
91       digitalWrite(LED_BUILTIN, LOW);
92       delay(1000);
93     }
94   }
95 }

```

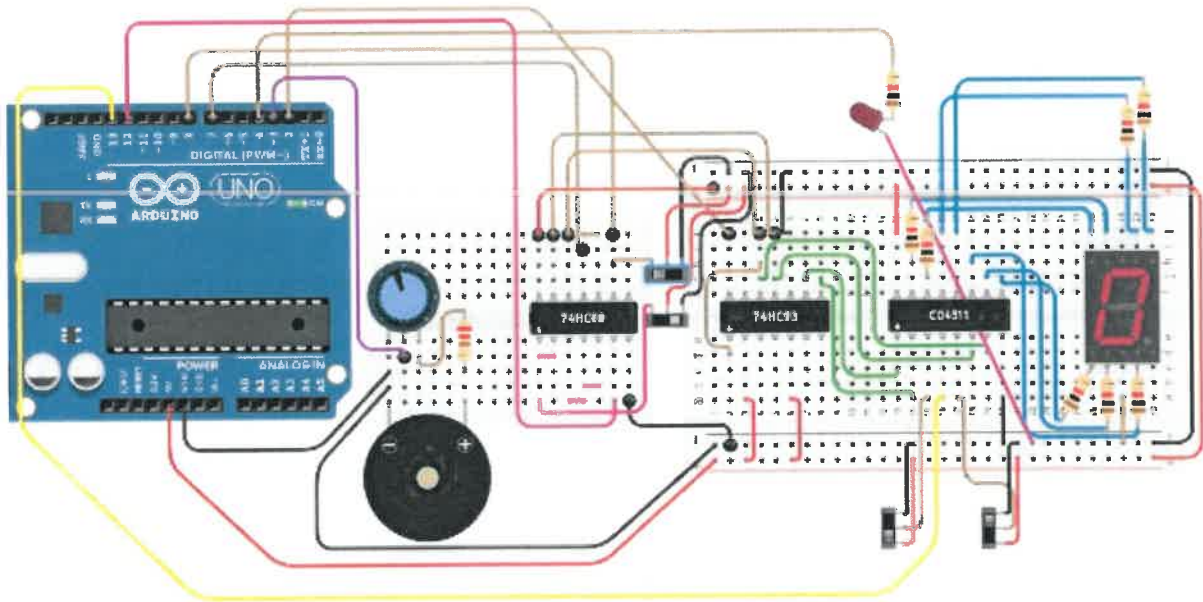
See lines 12, 29, 45, 77 and 90. Line 90 is evidently redundant, even in a while loop.

A problem with connecting the digital pin 13 is that the built-in LED of the Arduino (or LED_BUILTIN) applies primarily its status to what digital pin 13 is to change as, meaning that the 7-segment display will light up in response with the built-in LED. To counteract the issue into independent functioning, digital pin assignment for 'blanking' will allocate to digital pin 11, even if the pin involves Pulse-Width Modulation (that is an option to declare such function).

When the game starts with either win surface or wire contact being made, the 7-segment display does not light the LED until the waiting cycle and pre-game delay is complete with contacts still withdrawn.

The 7-segment display will also stop lighting up once all lives are lost in the game.

Impressive level of detail here.



The above image shows that digital pin 13 of the Arduino connects to pin 4 of the CD4511. The issue does not regard whether pre-game command or game loss occurs and will be resolved by moving the yellow wire to connect to pin 11 instead of pin 13. ✓

Pin 3 of the CD4511 will be wired with a power rail and pin 5 will be wired with a ground rail.

More Tones to Play and Grace Time Reduced

Hook-wire contact grace timing is now reduced from 1.2 seconds to 500 milliseconds.

$300 \text{ milliseconds} \times 2 \text{ cycles} \times 2 \text{ LED output combinations at a time} = 1.2 \text{ seconds}$ ✓

$1.2 \div 0.5 \text{ seconds} = \text{Dividing times by 2.4}$

$300 \text{ milliseconds} \div 2.4 = 125 \text{ milliseconds for each LED form}$

```
if (digitalRead(bit_ouch) == HIGH && digitalRead(bit0) == HIGH) { // Contact Ouch
  Serial.println("ZAP");
  tone(3, 136);
  digitalWrite(attack, HIGH);
  cycle = 0;
  digitalWrite(attack, LOW);
  while(cycle < 2){
    digitalWrite(indctr, HIGH);
    delay(125);
    digitalWrite(indctr, LOW);
    delay(125);
    cycle++;
  }
  cycle = 0;
  noTone(3);
  Serial.println("Ouch.");
}
```

For each hook-wire contact, a 136Hz beep emits for 500 milliseconds with allowance of further hook-wire contact before both desist.

Said tone does not play when the last life is used during gameplay. ✓

For tones played for win and lose conditions, see next page.

Win Condition Tone 1

```

void loop()
{
    if (digitalRead(win_hit) == HIGH) { // Contact WIN
        Serial.println("Game Set A");
        cycle = 0;
        lock_event = 1;
        while(cycle < 5){
            digitalWrite(indctr, HIGH);
            tone(3, 392);
            delay(50);
            tone(3, 528);
            delay(50);
            digitalWrite(indctr, LOW);
            tone(3, 659);
            delay(50);
            tone(3, 784);
            delay(50);
            cycle++;
        }
        cycle = 0;
        noTone(3);
        while(lock_event == 1){
            Serial.println("Win. It's working somehow.");
            digitalWrite(LED_BUILTIN, HIGH);
            delay(1000);
            digitalWrite(LED_BUILTIN, LOW);
            delay(1000);
        }
    }
}

```

When the hook touches the winning surface, the lines of code to the left renovate where it only makes the LED flash in five 200 millisecond complete intervals, changing state at 100 milliseconds at a time.

It is now presented alongside a 4-note C major (C, E, G; in here it starts with G) ascending arpeggio (where a chord of notes play individually in a sequence, ascending or descending) that plays five times in 200 millisecond intervals.

Note that the LED's flashing still completes in a parallel sequence with the buzzer playing in 50 millisecond intervals per tone.

In a future development, a more complex series of tones might play instead of an arpeggiating chord.

Lose Condition Tone 1

```

if (digitalRead(hal_touch) == HIGH || digitalRead(batc) == LOW) { // Contact Game Over
    Serial.println("Game Over. B");
    digitalWrite(see_crunk, LOW);
    cycle = 0;
    lock_event = 1;
    tone(3, 131);
    while(cycle < 3){
        digitalWrite(indctr, HIGH);
        delay(100);
        digitalWrite(indctr, LOW);
        delay(100);
        cycle++;
    }
    cycle = 0;
    tone(3, 123);
    while(cycle < 3){
        digitalWrite(indctr, HIGH);
        delay(100);
        digitalWrite(indctr, LOW);
        delay(100);
        cycle++;
    }
    cycle = 0;
    tone(3, 131);
    while(cycle < 3){
        digitalWrite(indctr, HIGH);
        delay(100);
        digitalWrite(indctr, LOW);
        delay(100);
        cycle++;
    }
    cycle = 0;
    noTone(3);
    while(lock_event == 1){
        Serial.println("Fail. It's working somehow.");
        digitalWrite(LED_BUILTIN, HIGH);
        delay(1000);
        digitalWrite(LED_BUILTIN, LOW);
        delay(1000);
    }
}

```

While the code appears more complicated, each tone plays per cycle in notes of C, B then C (131Hz, 123Hz and 131Hz in order).

Each while-loop plays a tone and performs two cycles of the LED flashing in 200 millisecond intervals.

The tones stop playing once the fourth while-loop is reached.

A future development might involve shifting arrays to produce an arpeggio that sounds like falling. However, the procedure will be difficult to research and develop since consideration towards arrays and matrices might complicate the procedure.

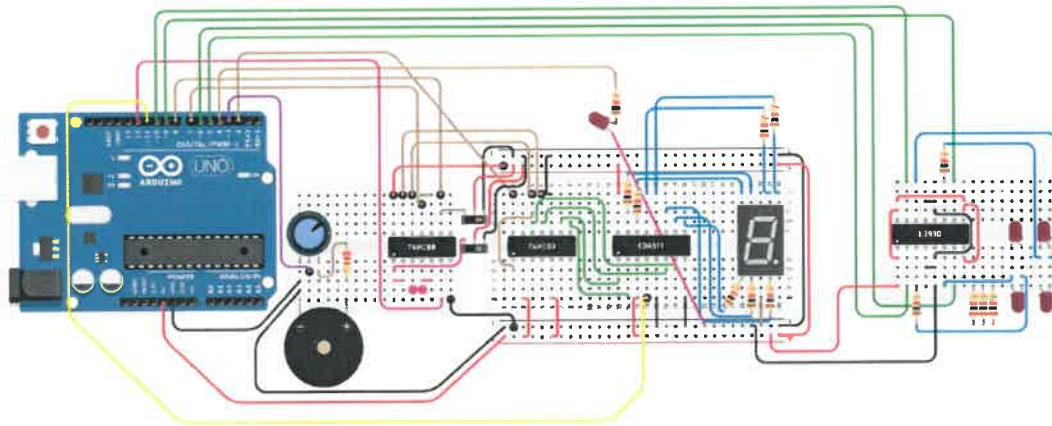
The next objective ensures that a stepper motor can be implemented to the Arduino.

Friday, 31 March 2023

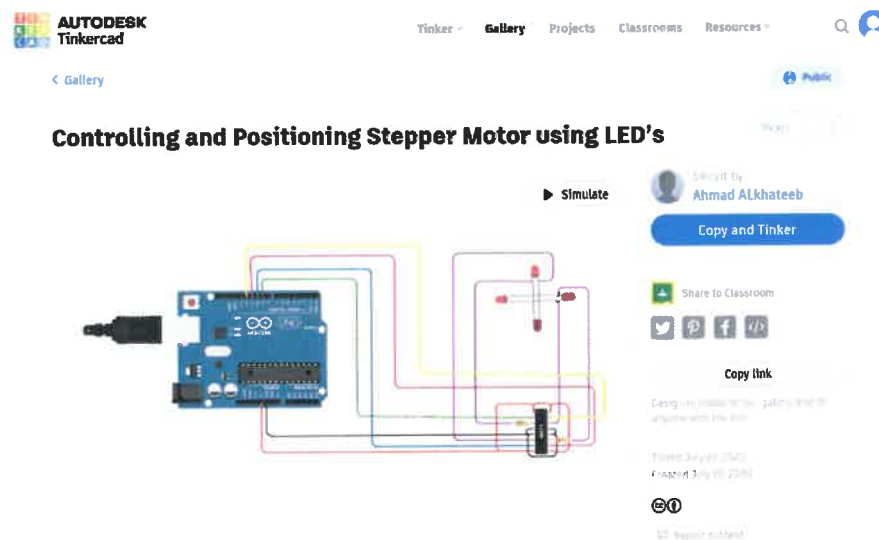
- TinkerCAD and C++ continuation.
- Rushed imitation of other users' submissions, instead taken as inspiration.
- Following the steps of another website for reference.

Stepper Motor Imitation Research

Because one of the current objectives is to implement a stepper motor, research to follow steps of other submissions end up complicating the procedure.



On the right side of the above image, there was an unprepared mean of imitating Ahmad ALkhatheeb to the project (see below). But further research finds that the ability to implement a Unipolar Stepper Motor can be performed on the current state of the personal project provided by Simple Projects (2018).



(ALkhatheeb, 2020)

For lines of code implemented to successfully simulate a stepper motor, see next page.

```

17 #define STEPS 17
18 int steA = 13; // Step A: all four steps function as if they interact with a coil.
19 int steB = 6; // Step B: each HIGH state represent an individual coil invoke
20 int steC = 5; // Step C
21 int steD = 8; // Step D: 'STEPS 17' as specified in case the stepper spins very fast.
22 Stepper stepper(STEPS, steA, steC, steB, steD); // Assigning pin variables as each coil.

33 int twiddle = 1;

34
35 void setup()
36 {
37   Serial.begin(9600);
38   pinMode(LED_BUILTIN, OUTPUT);
39   pinMode(bit_ouch, INPUT);
40   pinMode(bit_couch, INPUT);
41   pinMode(win_hit, INPUT);
42
43   pinMode(indctr, OUTPUT);
44   pinMode(attack, OUTPUT);
45   pinMode(see_count, OUTPUT);
46
47   digitalWrite(see_count, LOW);
48
49   while(digitalRead(bit_couch) == HIGH || digitalRead(win_hit) == HIGH){
50     if(digitalRead(bit_ouch) == HIGH || digitalRead(win_hit) == HIGH){
51       while(digitalRead(bit_couch) == HIGH || digitalRead(win_hit) == HIGH){
52         digitalWrite(LED_BUILTIN, HIGH);
53         tone(3, 880);
54         delay(400);
55         digitalWrite(LED_BUILTIN, LOW);
56         noTone(3);
57         delay(400);
58         twiddle *= -1; // Ungiven remark for if you want to turn wake a certain way.
59       }
60       tone(3, 105);
61       delay(1000);
62     }
63     digitalWrite(see_count, HIGH);
64     tone(3, 750);
65     delay(100);
66     noTone(3);
67     stepper.setSpeed(1);
68   }
69
70 void loop()
71 {
72   stepper.step(twiddle);
73   delay(200); // Test to see if "motor" does "spin".
74
75   if (digitalRead(win_hit) == HIGH) { // Contact WIN
76     Serial.println("Game Set A");
77   }

```

Lines 17 to 24, 50, 60, 65 and 66 have been added, and appear to function successfully. A fault being detected is that the LEDs that simulate the motor connects to the wrong legs.

Line 17 defines how much the stepper motor will spin. It is unknown whether the code functions similarly to `int variable = number`.

Lines 18 to 21 function as assigned digital pins on an Arduino to identify what parts of a coil to trigger. Note that a specific order on variables is induced since opposing coil triggering in a circular motion i.e: 'ACBD', is required to effectively turn the stepper motor.

That is because the coil assignment order in `Stepper stepper(x, a, c, b, d)` is laid out as: positive first coil pair; negative first coil pair; positive second coil pair; and negative second coil pair.

The coil pairs are on opposite sides in the sense of sinusoidal output.

If the logic is wired i.e: 'ABCD', then theoretically half or none of the rotation occurs since activating coils in parallel sides will lead to an unreliable speed of rotation.

Line 22 assigns how much the stepper motor turns and the four connections that resemble each coil of the stepper motor.

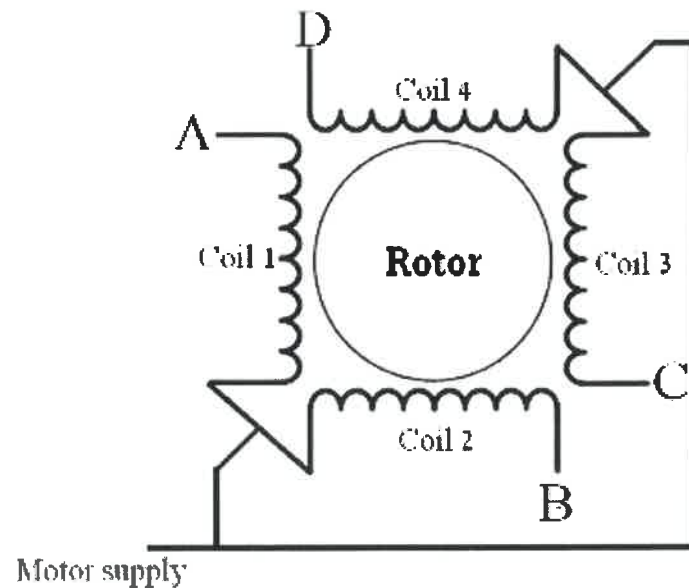
Line 24 counts as one defined step. Can be negative or positive from how Lines 41 to 51 end up configuring it. i.e: an odd number of stalling beeps in this state will invert the stepper motor to turn anti-clockwise.

Line 50 inverts the motor anti-clockwise if pre-game state is triggered via. conditions of Line 41.

Line 60 sets the rotation strength to 1, multiplied with the defined increment from line 17.

Line 65 physically rotates the stepper motor, and line 66 adds a delay in the process.

Usually the unipolar stepper motor has 5 wires one for motor supply and the other for coils. This motor has 4 coils and they are connected as shown in the figure below:



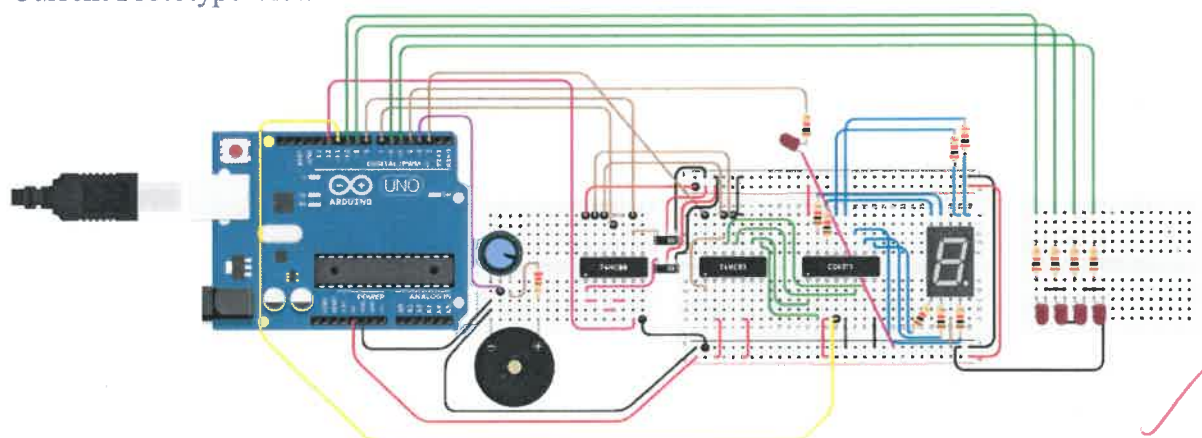
Unipolar Stepper Motor

(Simple Projects, 2018) ✓

Tuesday, 4 April 2023

- TinkerCAD and C++ continuation.
- Updated descriptions for Friday, 31 March 2023.
- Win and Lose Condition Property Modifications

Current Prototype View



For details on the modification(s) from Tuesday, 04 April 2023, see next page.

Win and Lose Conditions, Minor Modifications

```

100 void loop()
101 {
102     stepper.step(twiddle);
103     delay(10); // Test to see if "rotor" does "spin".
104
105     if (digitalRead(win_hit) == HIGH) { // Contact WIN
106         Serial.println("Game Set A");
107         cycle = 0;
108         lock_event = 1;
109         while(cycle < 5){
110             digitalWrite(indctr, HIGH);
111             tone(3, 392);
112             delay(50);
113             tone(3, 528);
114             delay(50);
115             digitalWrite(indctr, LOW);
116             tone(3, 659);
117             delay(50);
118             tone(3, 784);
119             delay(50);
120             cycle++;
121         }
122         cycle = 0;
123         noTone(3);
124         while(lock_event == 1){
125             Serial.println("Win. It's working somehow.");
126             digitalWrite(LED_BUILTIN, HIGH);
127             digitalWrite(see_count, HIGH);
128             delay(1000);
129             digitalWrite(LED_BUILTIN, LOW);
130             digitalWrite(see_count, LOW);
131             delay(1000);
132         }
133     }
134
135     if (digitalRead(bit_couch) == HIGH && digitalRead(bit0) == LOW) { // Contact Game Over
136         Serial.println("Game Set B");
137         digitalWrite(see_count, LOW);
138         cycle = 0;
139         lock_event = 1;
140         tone(3, 131);
141         while(cycle < 4){
142             digitalWrite(indctr, HIGH);
143             delay(100);
144             digitalWrite(indctr, LOW);
145             delay(100);
146             cycle++;
147         }
148         cycle = 0;
149         tone(3, 123);
150         while(cycle < 4){
151             digitalWrite(indctr, HIGH);
152             delay(100);
153             digitalWrite(indctr, LOW);
154             delay(100);
155             cycle++;
156         }
157         cycle = 0;
158         tone(3, 131);
159         while(cycle < 4){
160             digitalWrite(indctr, HIGH);
161             delay(100);
162             digitalWrite(indctr, LOW);
163             delay(100);
164             cycle++;
165         }
166         cycle = 0;
167         noTone(3);
168         while(lock_event == 1){
169             Serial.println("Fail. It's working somehow.");
170             digitalWrite(LED_BUILTIN, HIGH);
171             delay(1000);
172             digitalWrite(LED_BUILTIN, LOW);
173             delay(1000);
174         }
175     }
176 }

```

The win condition's code is appended so that the 7-segment display flashes in a reasonable interval, signalling a completed game.

The lose condition's code is modified so that the individual tones play twice as long i.e: from `while(cycle < 2)` to `while(cycle < 4)`. ✓

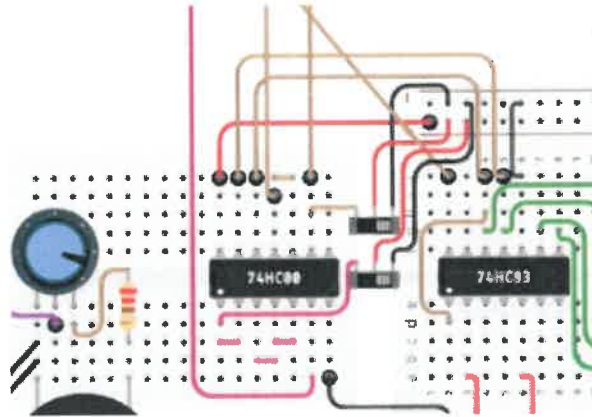
Line 66 is due to contain a smaller integer once physical prototype testing commences, and will decrease up to outright removal once testing confirms reliability and function.

While in a while-loop, the wire stops spinning as it is involved in a condition that does not follow inside the outermost `void loop()` bracket i.e: program not focussed with the code layer containing line 65 that rotates the wire.

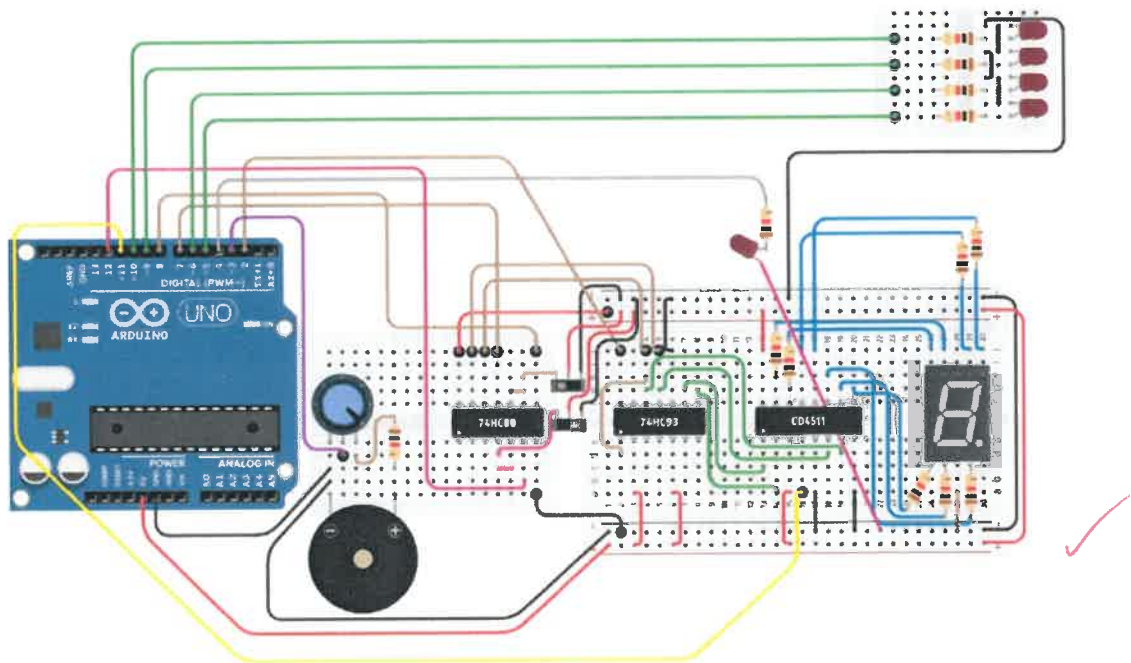
Wednesday, 3 May 2023

- Modification of both TinkerCAD prototype and its code to allow appropriate use and decoding of 74HC00 outputs.
- Provided proof of Texas Instruments component use e.g: 74HC00 IC on prototype.
- List of details that have been removed due to time constraints and external matters.

Renovating for the 74HC00



Previously, the 74HC00 used to produce digital outputs occurred to have unreliable configuration in the wiring (see above).

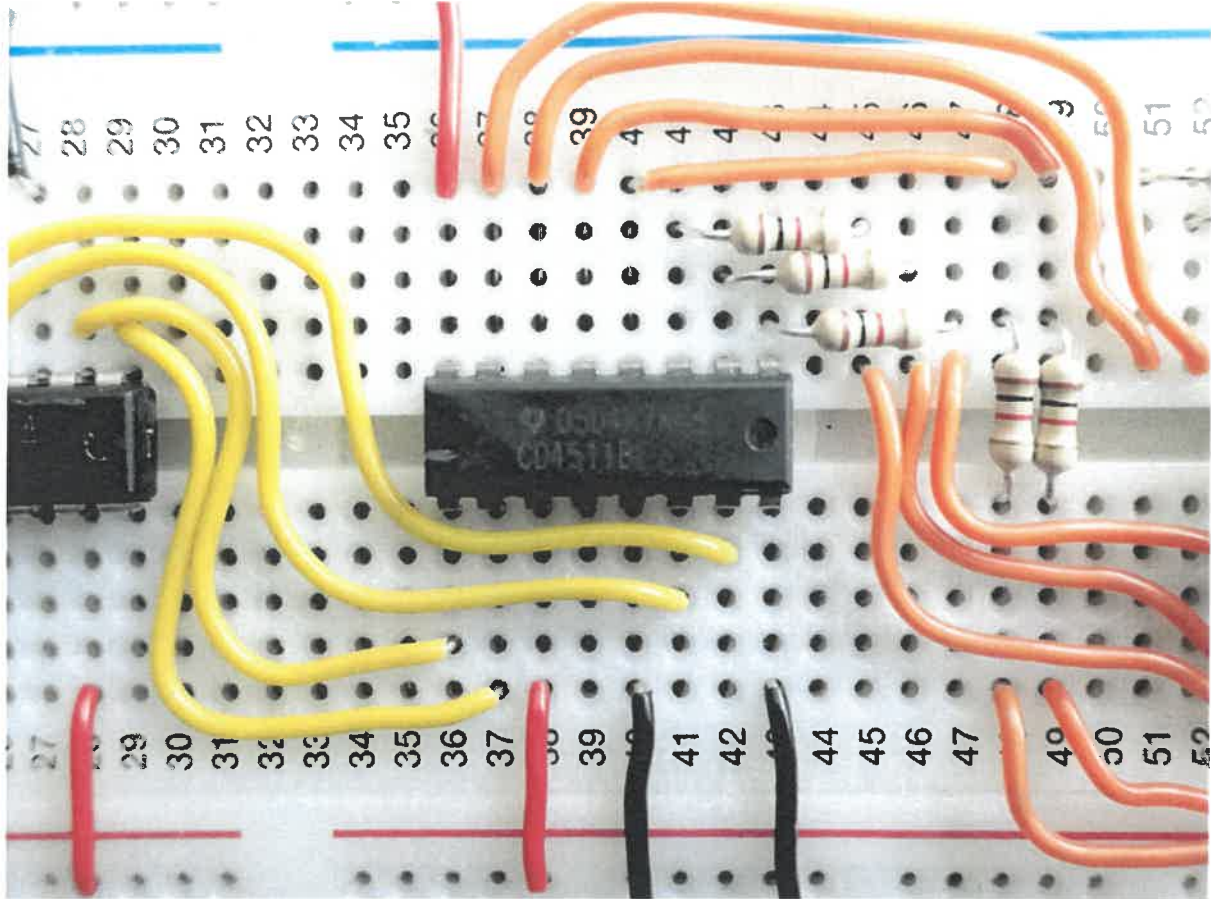


The result of the gradually large change is so that pins 8 to 10 are properly configured to be occupied by their corresponding wiring. This ensures that the 74HC00 is properly assigned with variables that are due for inversion and are read by the Arduino's program with respective boolean states.

For context with the top-right breadboard in below image, it should be indicative for a ULN2003A IC.

For code in relation to current contexts, refer to Project Appendix 5.1.

Proof of Texas Instruments Component Usage



As is read from 'Texas Instruments - Important Notice and Disclaimer', if components containing identification of Texas Instruments is in use, there can be allowance to reference with their datasheets with respective components. However,

List of Removed Details 1

- Where there was said to be DIP switch settings for features such as: timer incrementation or set times for games, etc.... Time constraints and external matters prevent the implementation of such from happening.
- Win and lose condition tunes were planned to include more complicated compositions. And with the same reason as above, the tunes are kept to simple lines of code. Subjective exception in that the commands use 'for loops' while monitoring the external LED that flashes (and might have to be removed alongside).
- 74HC08 ICs (or quad-AND gate ICs) and 74HC32 ICs (quad-OR gate ICs) were requested, but no use was given for them. The same applies with the 74LS-series ICs as of redundancy in parameters.
- Green, blue and violet/purple wires were not used due to having 'stranded core' as opposed to the specified 'solid core' which allows reduced mess with equipment.
- There was a sketch of an instruction manual in the sketch prototype made in Microsoft Paint. But plans on creating an instruction manual is to be removed as is by the same reasons as the first two bullet points.

-Good reflection.

Thursday, 4 May 2023

- + Added image that proves that Texas Instruments components are used to properly use respective component reference datasheet for Wednesday, 3 May 2023.
- Brief tutorials for a simple product design, namely a rod, featuring a schematic design beforehand. Design Rendering is also included as a tutorial.

Getting to Fusion 360

Permission, access and development on Fusion 360 is decided for the project.

Refer to Project Appendix 1.1.1 for instructions.

Basic Steps to Schematic Design in Fusion 360

Refer to the following chapters from Project Appendix:

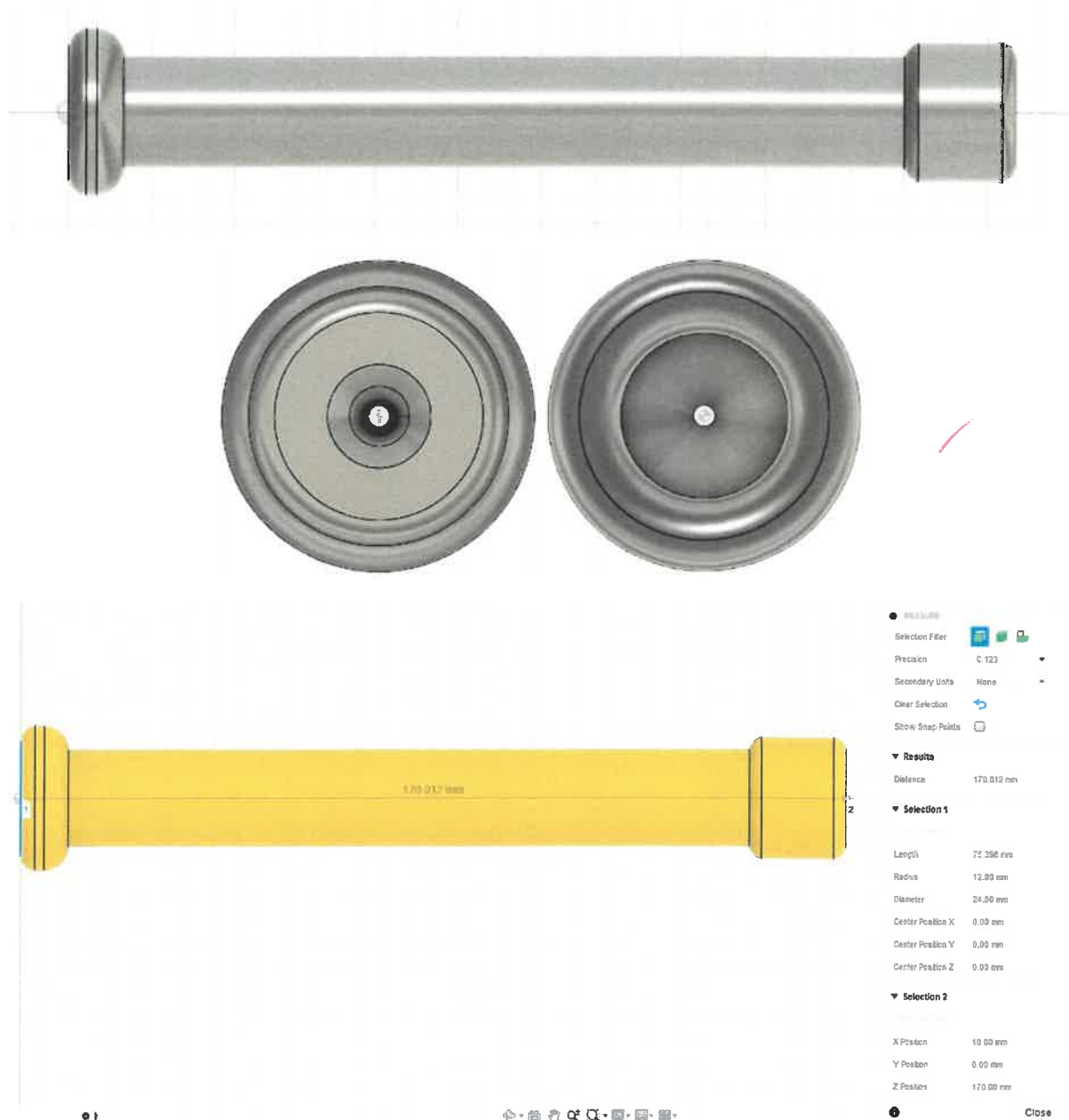
- 1.1.2 for viewing all Stock Libraries in Fusion 360.
- 1.1.3 for producing a simple rod shape where thin cable can be inserted through – the main product for the tutorial.
- 1.1.4 for rendering a design in Fusion 360.

Friday, 5 May 2023

- Creation of new rod, discarding one developed at 'Thursday, 4 May 2023'.
- New tutorials for rod editing.

New Rod Design 1

Considering the perceived simplicity of a rod design, another rod is produced for ergonomic and slightly improved purpose.



How to Change Physical Material of Rod

Refer to Project Appendix 1.1.5 for changing physical material for a design, i.e: a rod in Fusion 360.

The next objective is to create a stand for the wire.

-This is good evidence!

-Shows the work you are working to.

Wednesday, 10 May 2023

- Creation of Wire Stand so that the wire can rotate while the game starts.

How to Create a Wire Stand that should Rotate with the Step Motor

Refer to Project Appendix 1.1.6 for instructions.

Friday, 12 May 2023

- Continuation of 'wire stand' tutorial from Thursday, 11 May 2023.
- Arduino practice as code from TinkerCAD is sent to Arduino IDE.
- First prototype test, expected to contain bugs in the wiring.
- Extra notes regarding the project.

Uploading TinkerCAD Code to Arduino UNO

Refer to corresponding chapters of Project Appendix with below tutorials.

1.2.1 for testing an Arduino UNO by demonstrating an in-built program on it.

1.2.2 for transferring TinkerCAD code to an Arduino UNO.

First Physical Prototype Test

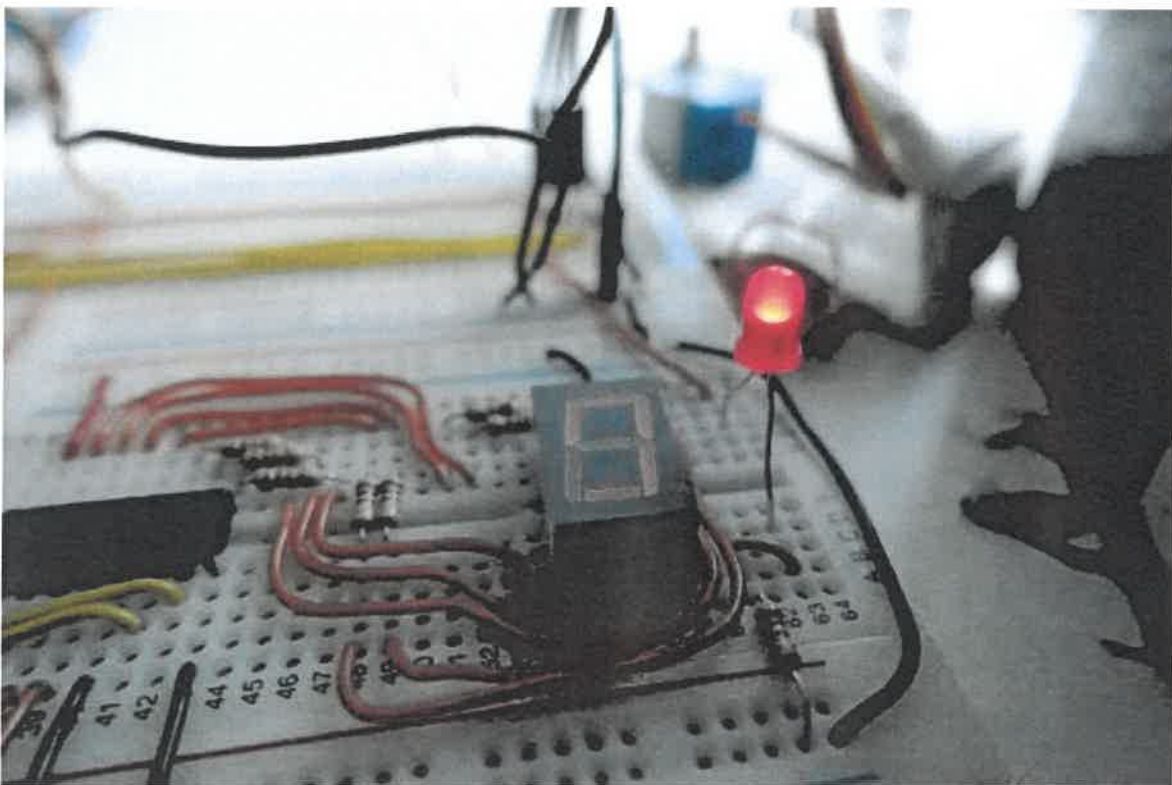
The wiring on the physical prototype leaves many room for improvement since the transcribing technique is less prepared and planned.

Once verified and uploaded, the game fails to work overall.

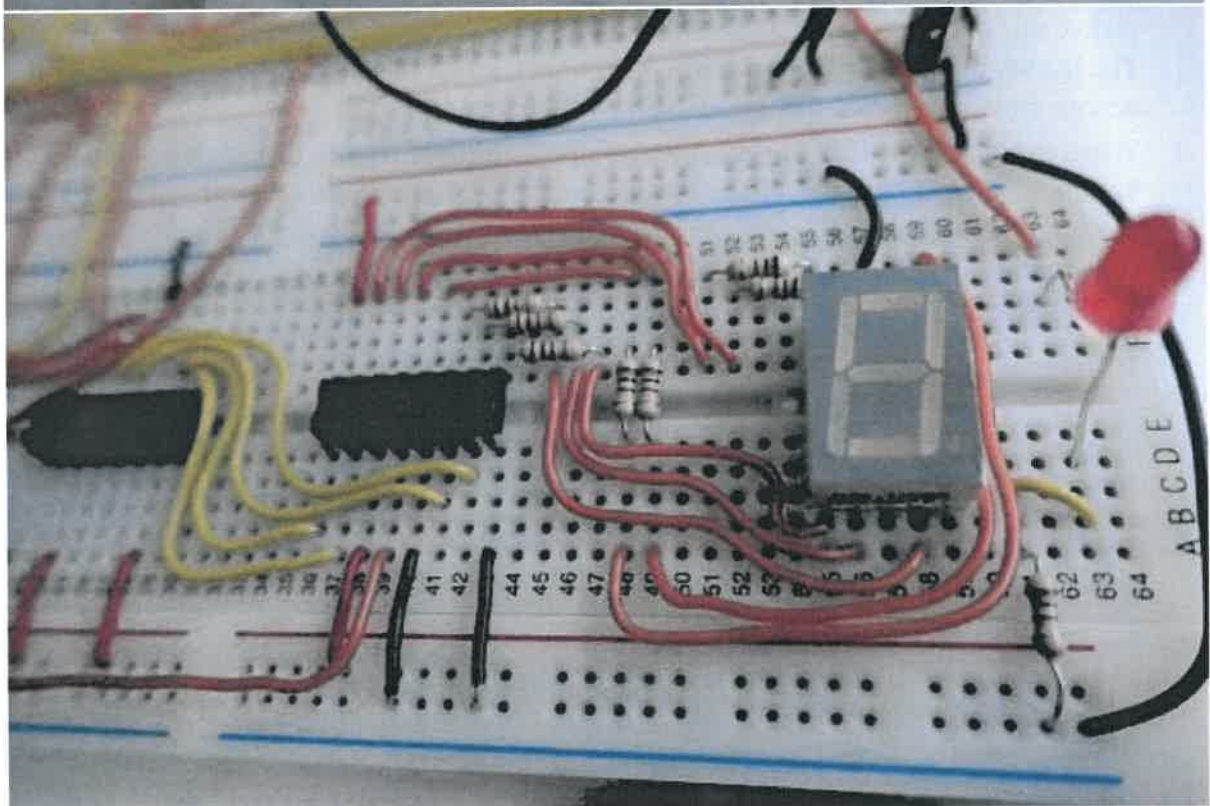
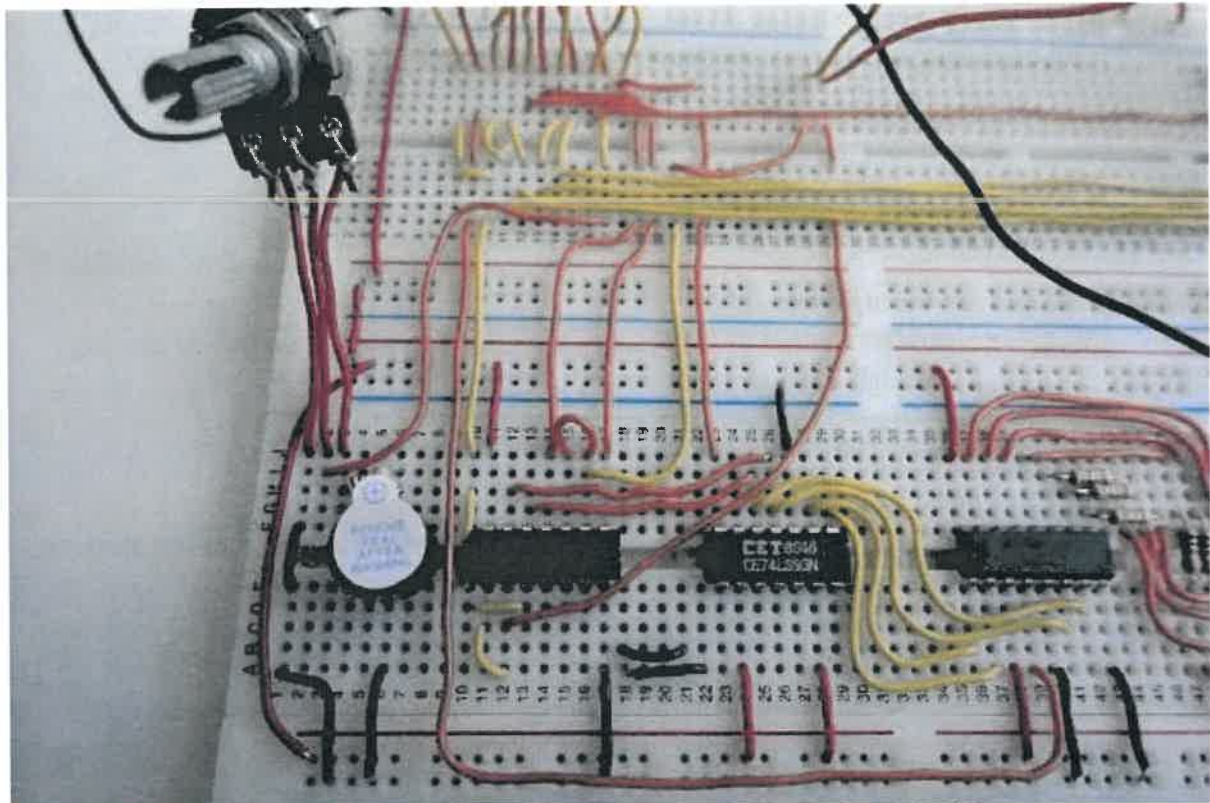
The list of events are as recorded during the first physical prototype tests.

- The buzzer did beep, but was barely audible once its sticker was peeled off since the most resistance it ran through was $1k\Omega$.
 - Added another of the resistor in parallel to reduce resistance running through the buzzer.
 - All but the fail-state tune sequence played.
- The 7-segment display briefly fully light except for segments 'E' and 'G'. But afterwards becomes dim. It does flash.
 - It did indicate that the program is not properly wired via. digit not counting up.
- The LED beside the 7-segment display lights up fully.
- The two switches did work on their own with power supplied. However, the hook-wire contact switch only caused a sound to play. The 7-segment display did not change.
- The stepper motor was not sensed to be spinning, even if the `delay()` line below its invoking line was removed.
- The pre-game condition did not get sensed i.e: intended lines in `void setup()` are not picked up. The game starts without having to prompt the need to set switches to default. The buzzer did not play its pre-game error tone nor had the project prevented further actions from switches set to non-default states.
- When the game is at its win-state, the Arduino's 'L' light flashes in a square-wave pattern but the 'TX' light pulses in miniscule intervals in the same cycle as the 'L' light.

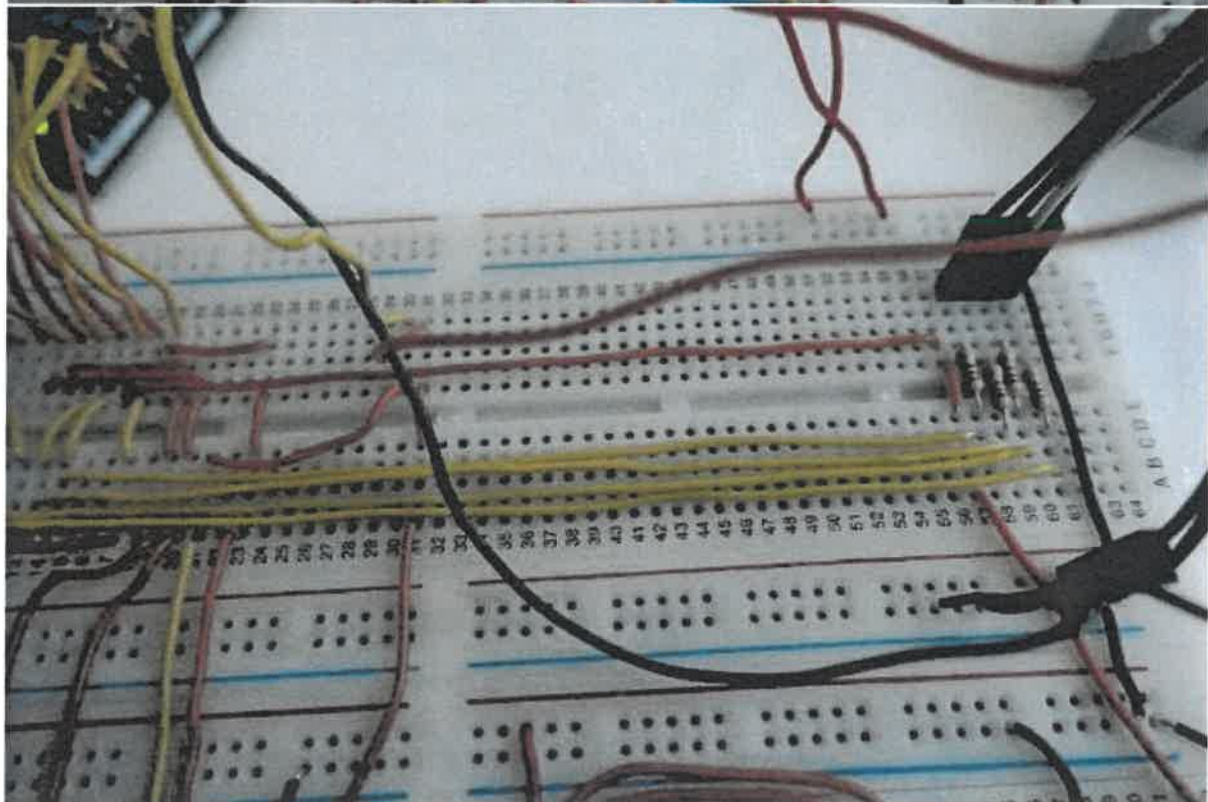
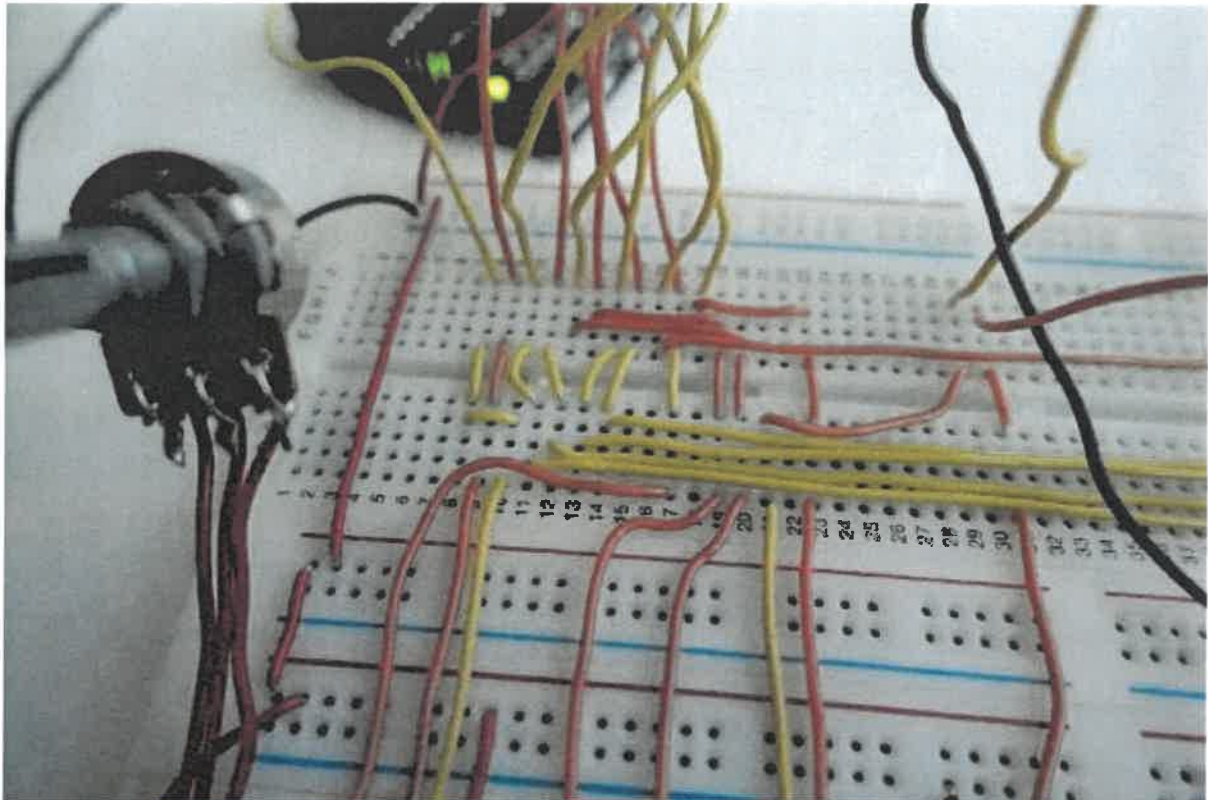
— Excellent problem solving.

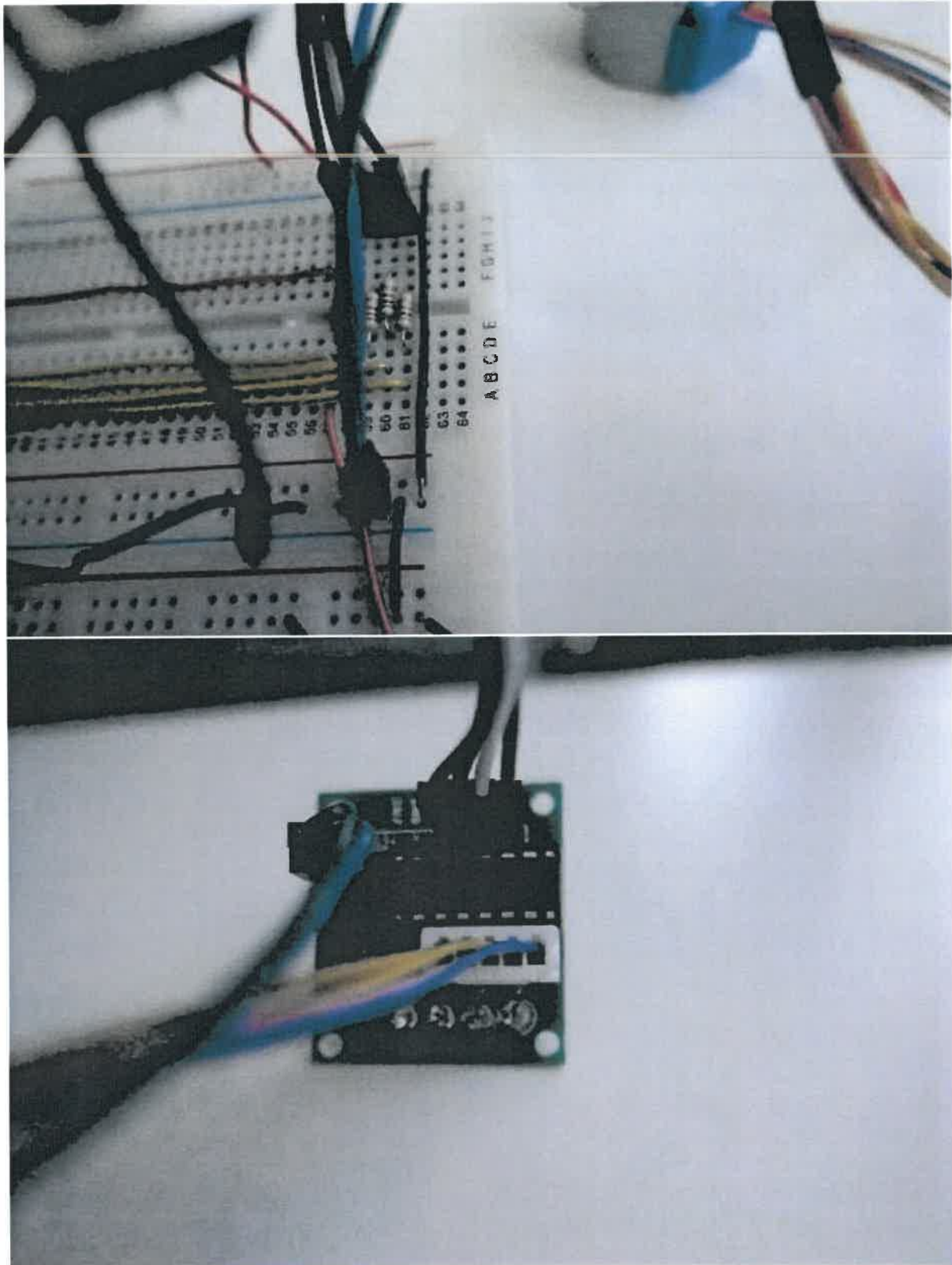


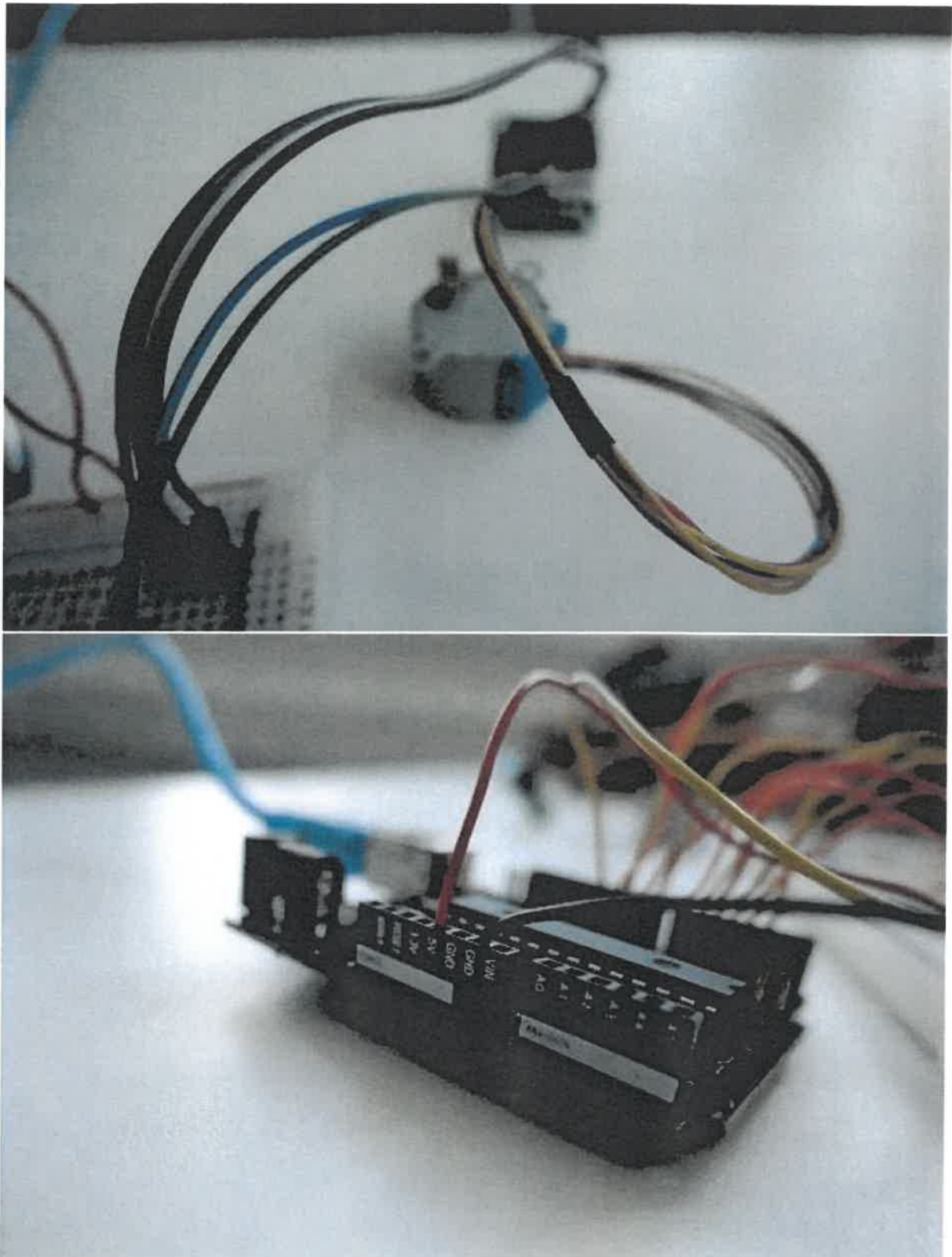
For more images of the current state of the breadboard, see next pages.

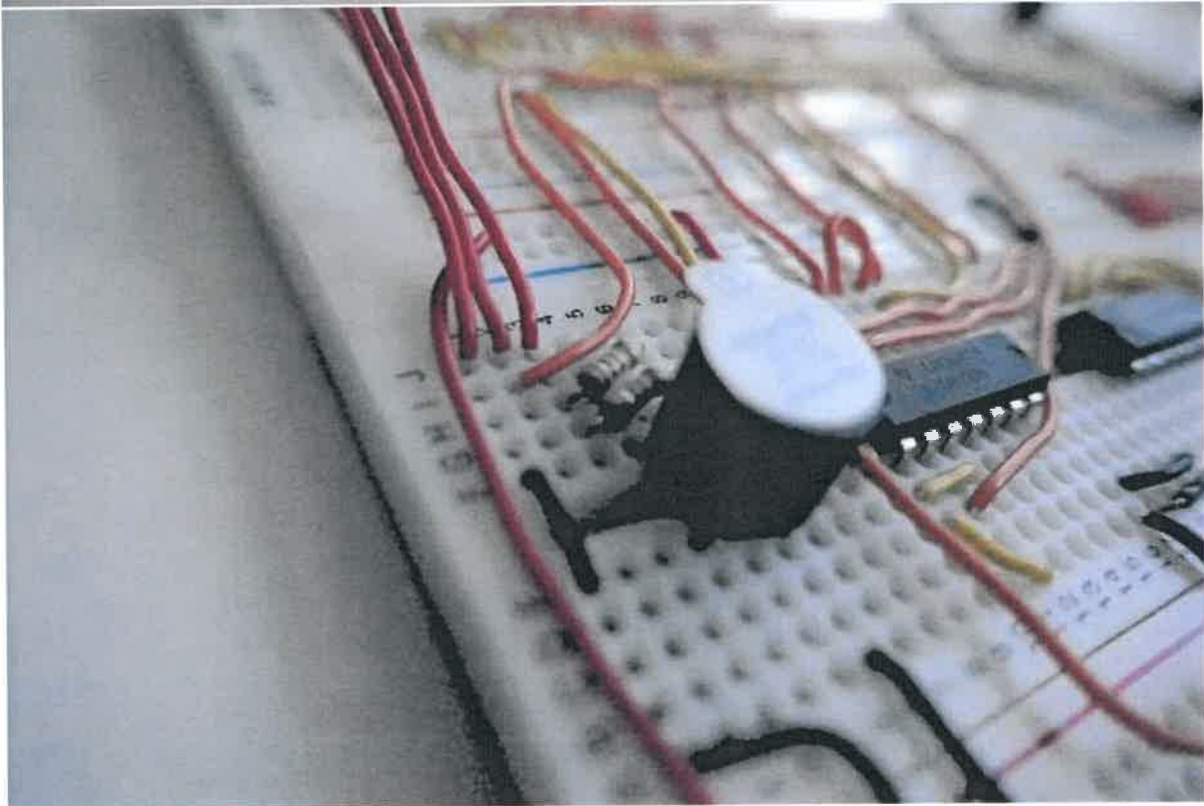
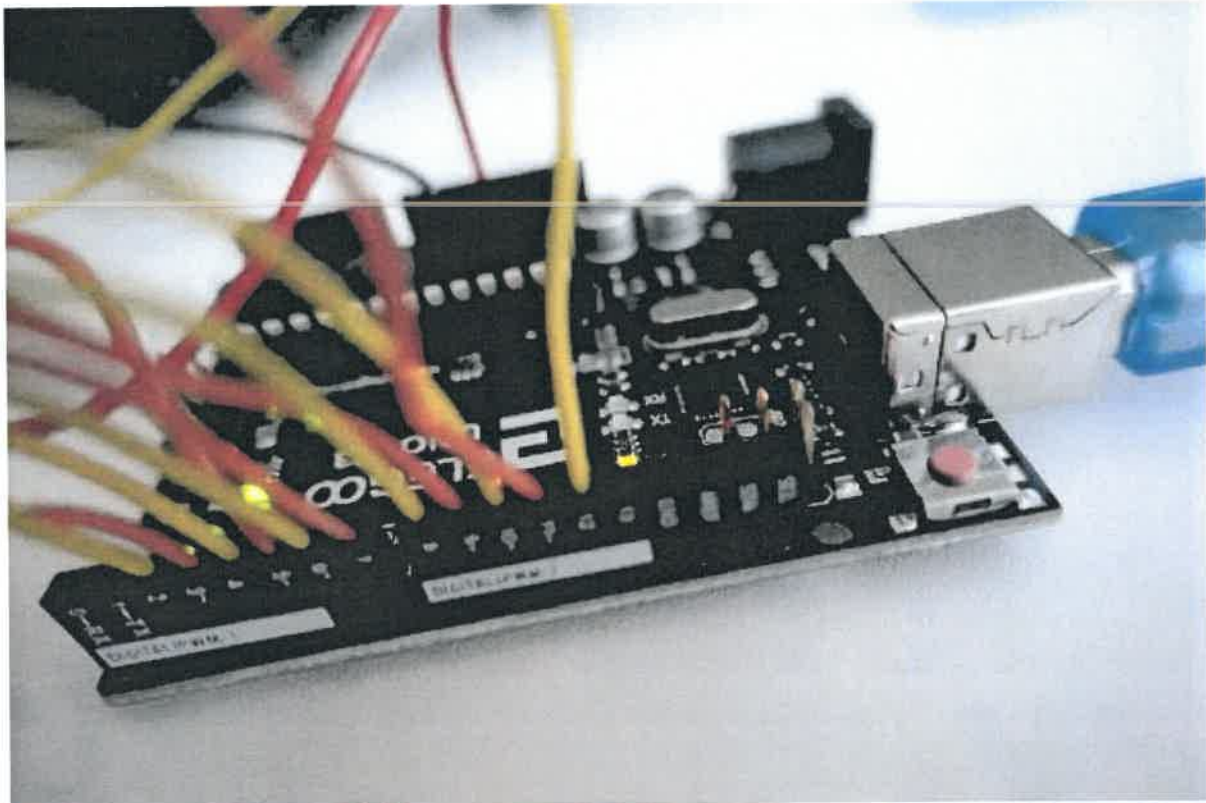


Captions!









Monday, 15 May 2023

- Arduino Code to be edited due to error upon inputs not being sensed pre-game and that the stepper motor is not rotating. Plus the wiring is flawed as the counting is not carried out.

Faults at Wiring and Code Modifications

Because the initial flaws are as stated from Friday, 12 May 2023 – ‘First Physical Prototype Test’, more wiring is carried out as well as some modification of code.

For code in 15 May 2023, refer to Project Appendix 5.2.

The wiring from the 7493 IC has not got a wire that connects from pin 12 to pin 1 presumably resulting in that the counter does not count up.

New wiring for segment ‘E’ (and possibly ‘G’) of the 7-segment display is to be carried out so to test if the segments are back working.

The step variable is set to 50 and is removed of continuous delay.

New code will be replacing and uploaded so to determine whether the concerned aspects are working properly.

See next page for the next two prototype tests with said code implemented.

Two Physical Prototype Tests

The second test results are as reported:

- The step motor still does not spin.
- The 7-segment display still does not light up its 'E' segment nor its 'G' segment.
- Reports may have to conclude in this test that too much resistance is obstructing the current flow that components fail to function as intended.
 - Especially when a 10k potentiometer might be the result of too much resistance occurring through the circuit. Thus might have to replace the potentiometer for a one with smaller resistance.
- Another bug spotted is that pin 6 of the 7493 IC is connected to voltage rail instead of pin 5. Quickly fixed.
- + Beeper works as intended.
- User inputs are possibly bugged with poor component switches.

The third test produces the following results, from :

- The step motor does not spin.
- The 7-segment display began to light up more than previously, but the counter is set to displaying '1' instead of '0'. All but segments 'E' and 'G' are confirmed positive. Segment G still remain ambiguous.
- Potentiometer replacement to perform due to current flow concerns.
- + Beeper works as intended.
- User inputs are still bugged by the switches and potentiometer.
 - Events occur only when the potentiometer slides nearest to its LOW state.
 - Results against the 7-segment display vary between input permutations, but are not logged due to time constraints.

List of amends to carry out:

- Acquire a potentiometer of smaller resistance.
- Acquire Single-Pole Double-Toggle switches for breadboards as the larger PCB-mount (printed circuit board) switches should be used for circuit boards.
- Acquire new 7-segment display of Common Cathode variant.
- Explain why adding (pin) connector components to the circuit design benefits production. Provide explanation on document with evidence from Fusion 360.

In addition, explain why pin numbers are not as neatly organised and announce if amendments are to be made about it.

Wednesday, 17 May 2023

- TinkerCAD to Arduino continuation as partial acquiring of components listed in 'Monday, 15 May 2023'.

Breadboard 7-Segment Display Tester – Common Cathode

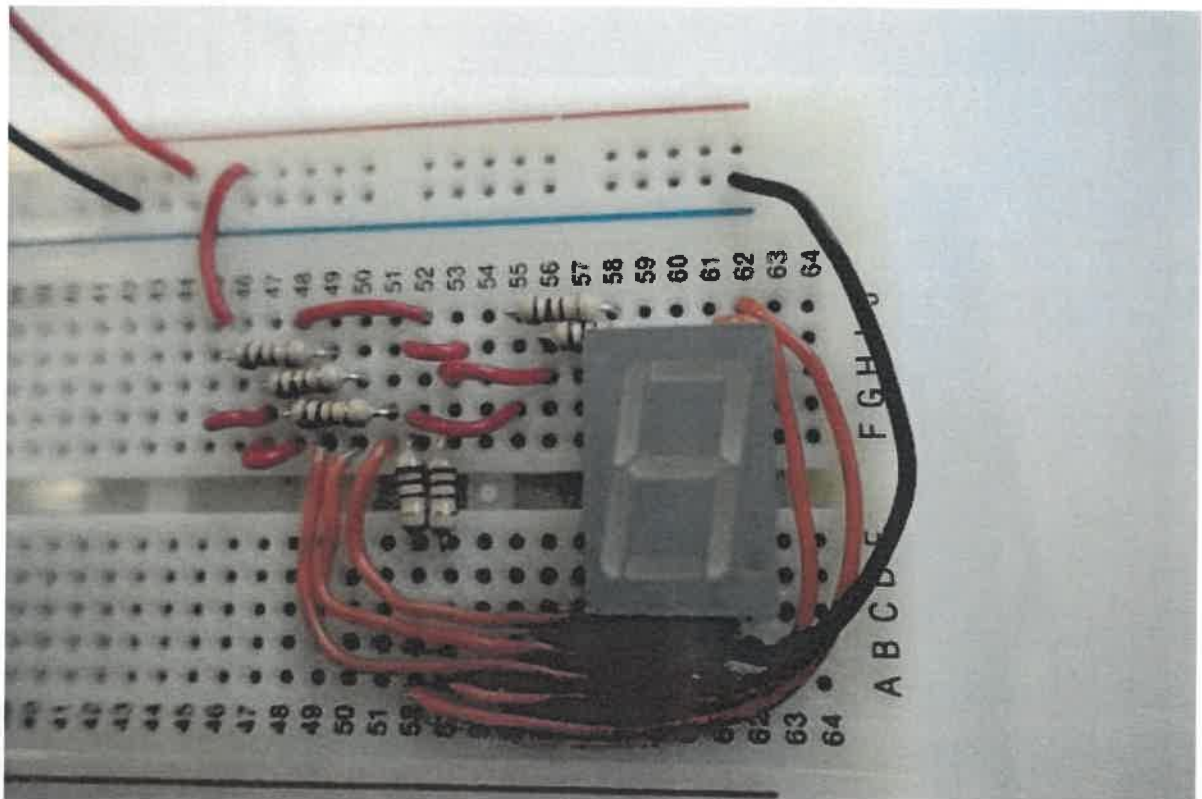
Having taking new 7-segment displays, smaller potentiometers and slide switches, the new 7-segment displays acquired turned out to be 'common anode' due to logistic rush without realising provided acquisition of extra 'common cathode' 7-segment displays.

A 'common cathode' 7-segment display is roughly tested for each segment with a $1k\Omega$ resistor before the pins. The 7-segment display is confirmed positive for damageless indication and will replace the existing 7-segment display from the main prototype. ✓

Whilst testing the breadboards used for the 7-segment display testing, and to primary extent the prototype, the breadboards are then diagnosed that the halves of the breadboards (not the IC-spaced separator) require power and grounds in between because the breadboards used are split; perhaps so to allow different power values per segment of the breadboard. (Maxfield, 2022) ✓

Therefore, the 7-segment displays are confirmed to be not at fault, but as indicators for faulty circuitry. *Good use of literature.*

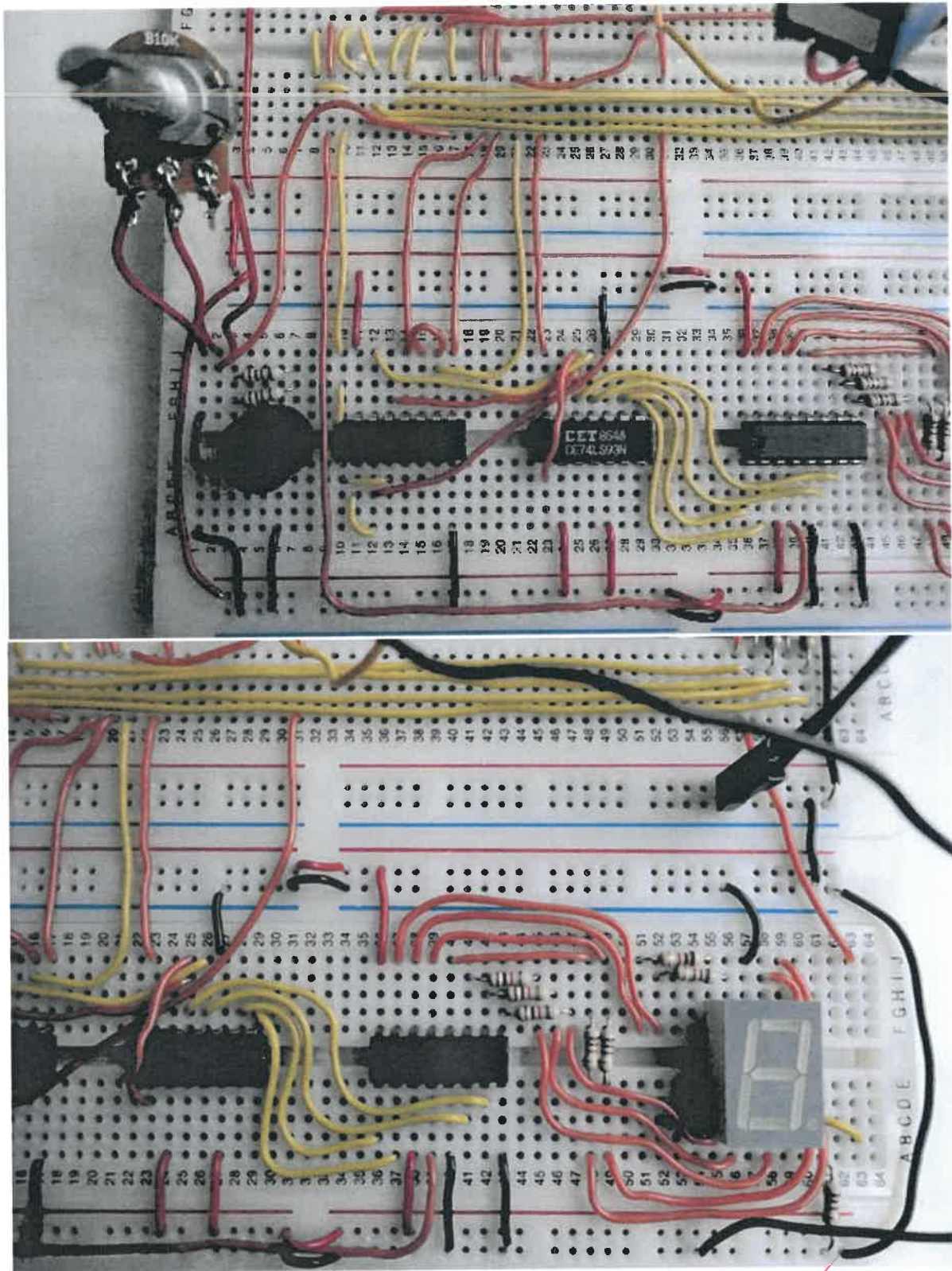
Image for the makeshift 7-segment display tester is as provided below: ✓



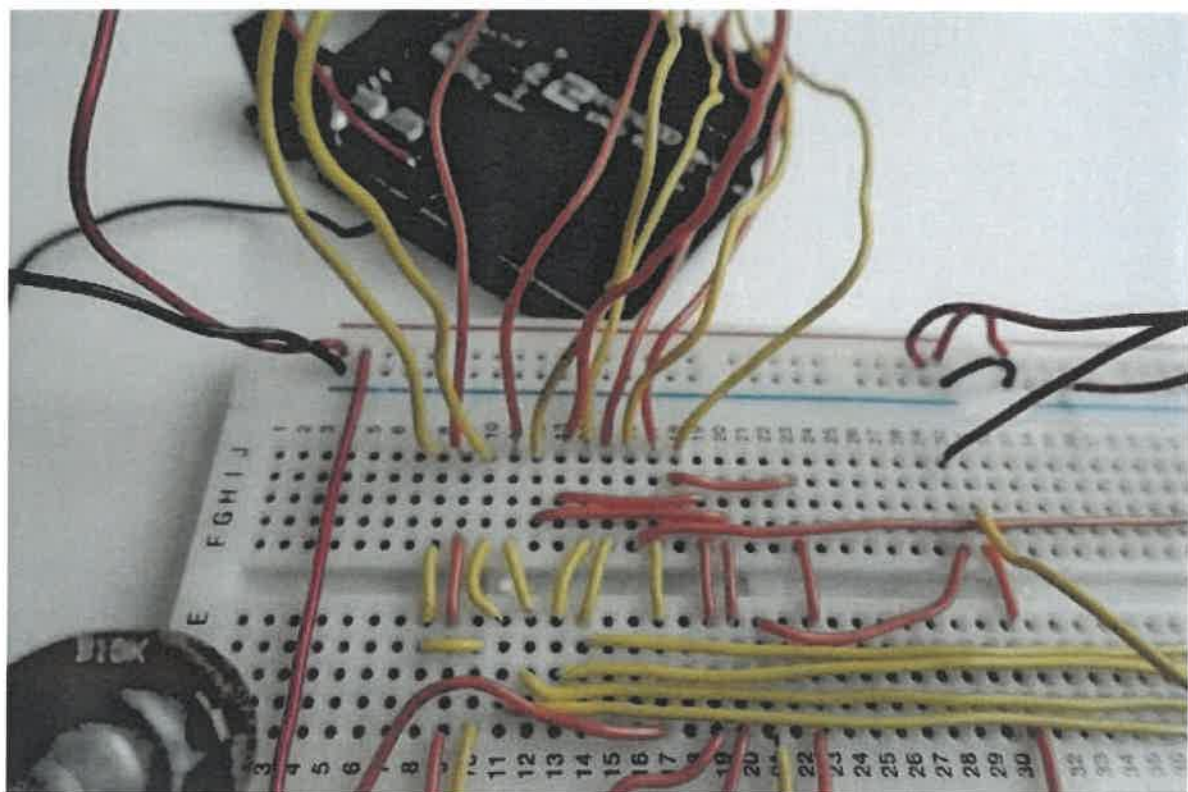
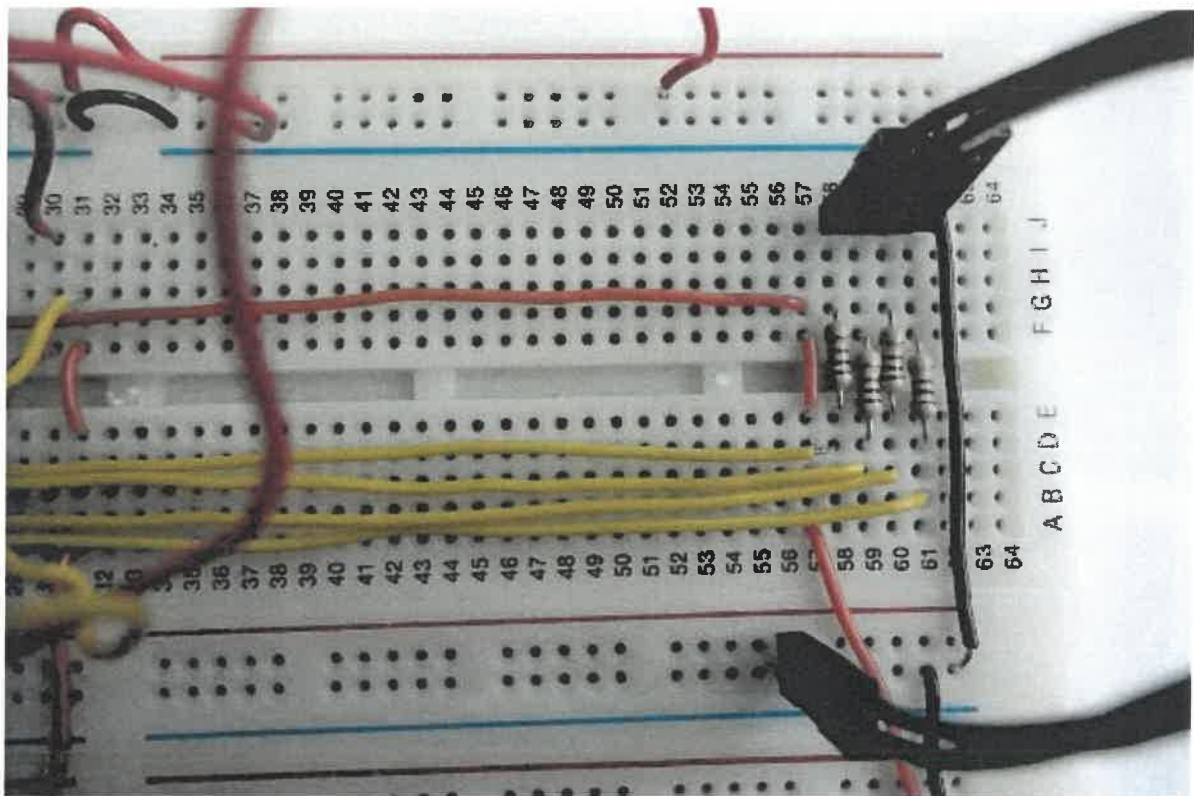
Good improv.

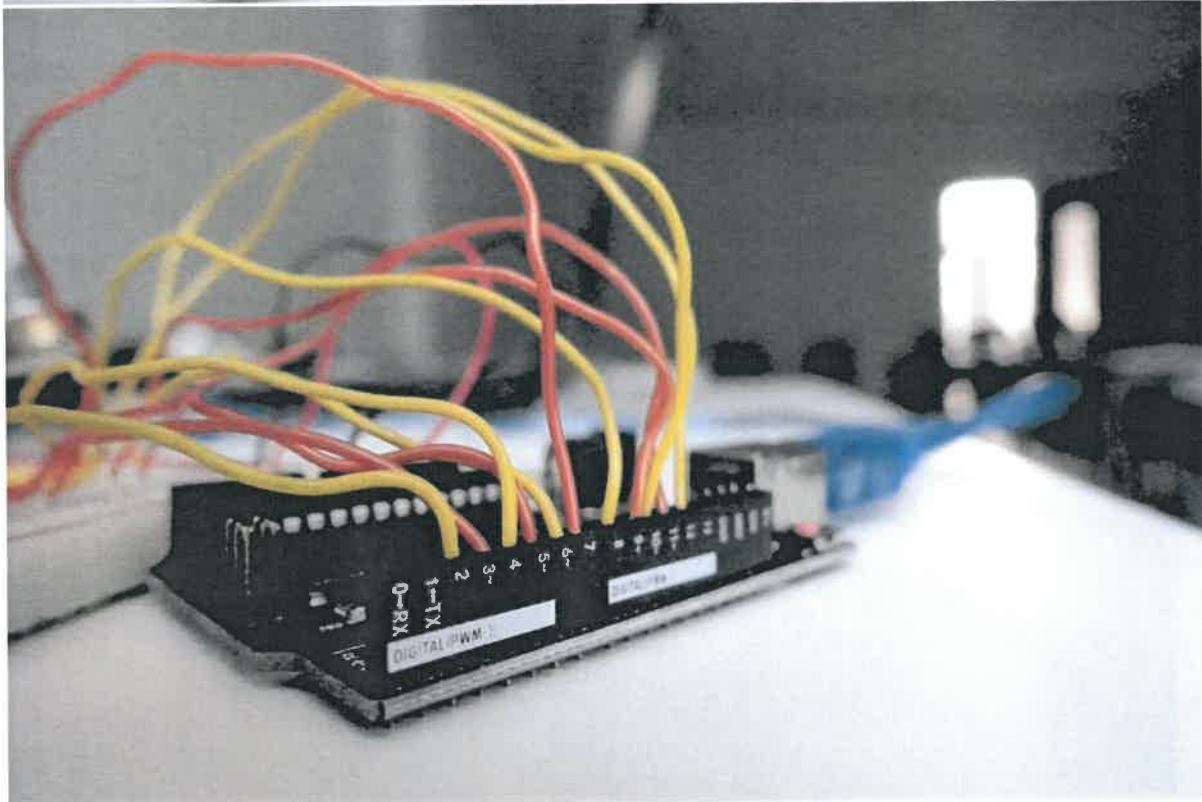
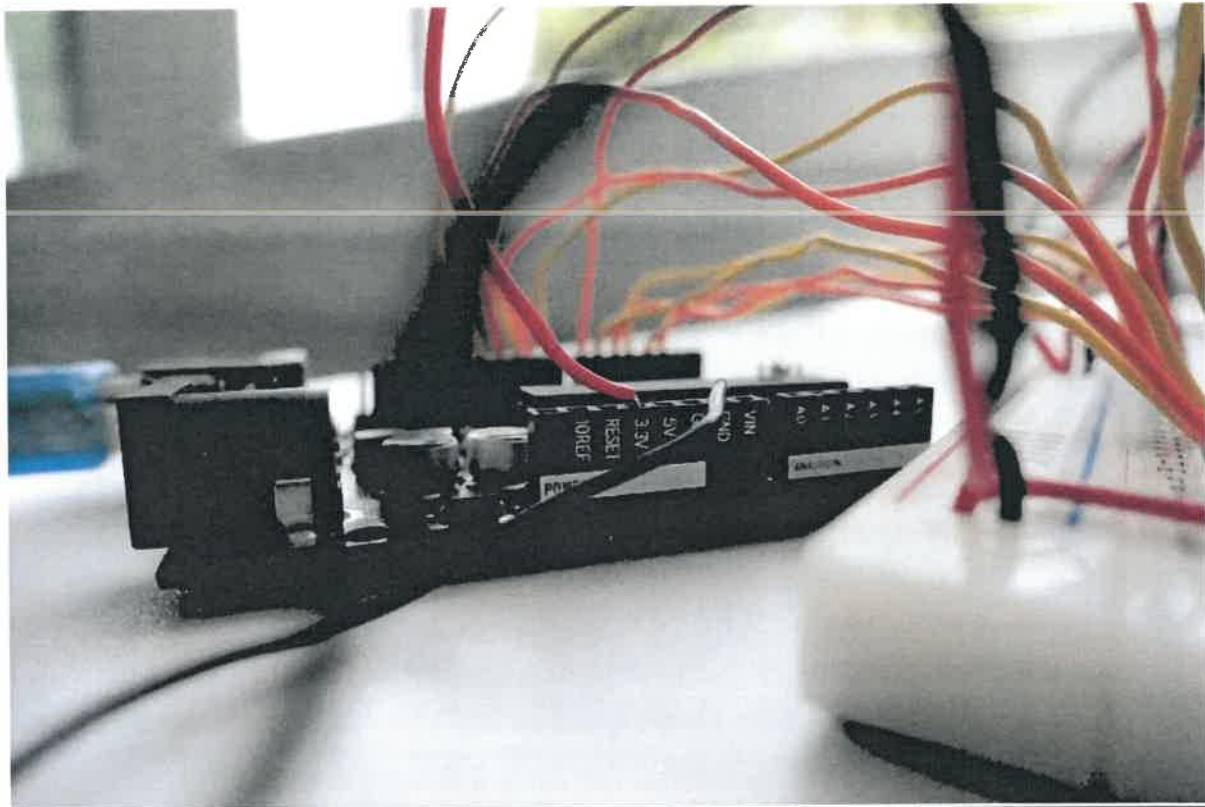
Breadboard Prototype Test Number 4

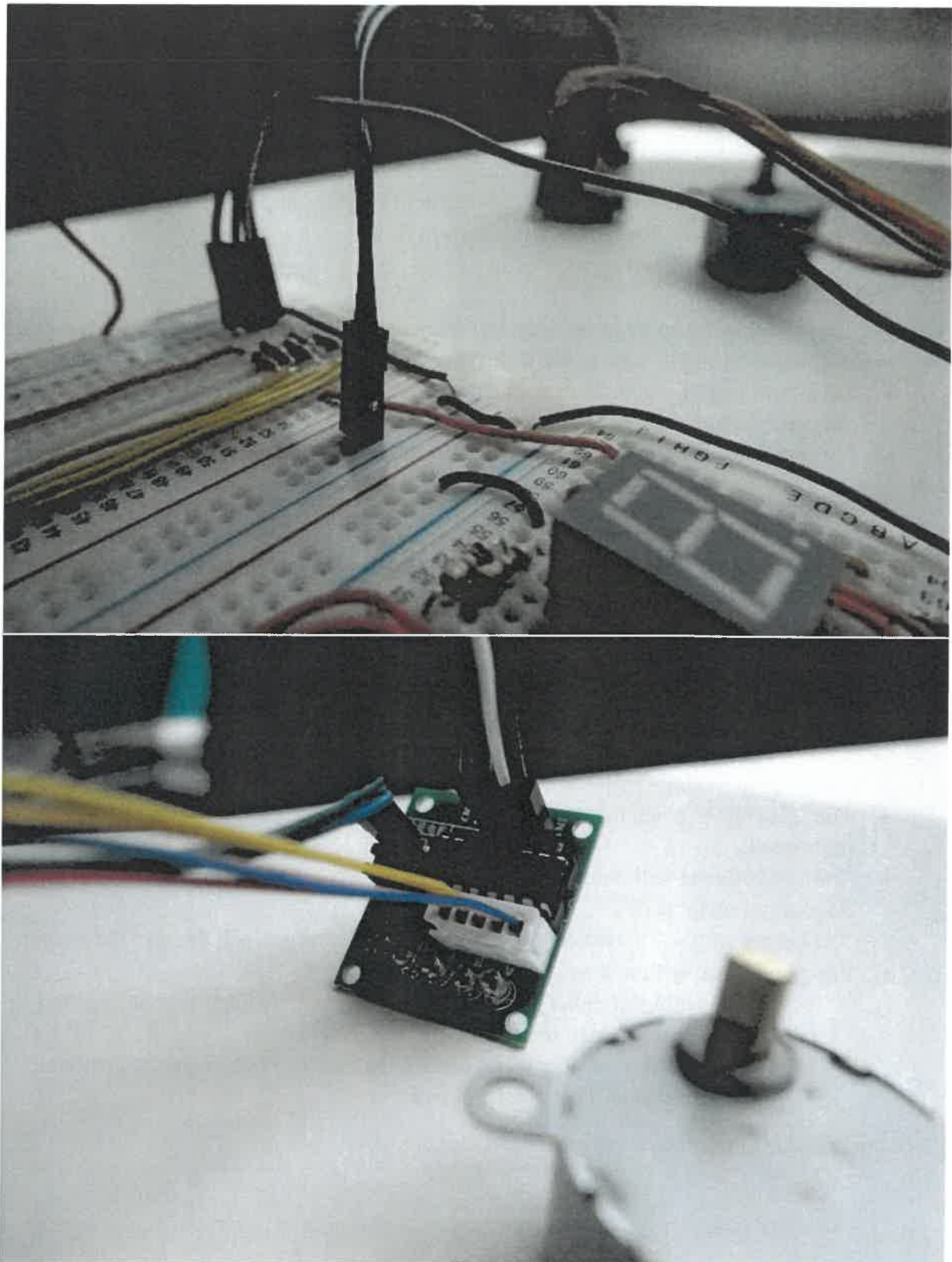
Images of the next prototype iteration are as displayed below:



Unviewed Append: Extra red wire to connect to the uppermost power rail.







The following changes to the prototype are induced:

- One makeshift toggle switch is replaced with a quality toggle switch due that the second acquired toggle switch ended up having a detached input leg.
- Bridging for power and ground rails are implemented on the breadboards for assured circuit flow.
- Replacement of 'common cathode' 7-segment display, despite prior component test confirmed without destruction from current flow.

Results of test number 4 are as stated:

- The 7-segment display lit up fully, but there are no beeps emitted by the beeper due to misplaced wiring to send to the positive pin of the beeper.
- No adjustment of inputs other than a non-working potentiometer being briefly adjusted without change of results.
- Forgotten to re-attach the blinking LED. Immediately re-attached for test number 5.

The result of not supplying a power and ground rail for ICs can result in anomalous functions and outcomes. Especially with the CD4511 IC (the third IC in the prototype) where e.g: the immediate start to '0001' translated as '1' from the 7-segment display with a barely dimmer lighting than intended.

Breadboard Prototype Test Number 5

Results of test number 5 are as stated:

- The stepper motor finally rotates.
 - The stepper motor is seen to rotate very slowly, so an edit of its rotation speed is to increase its RPM from '#define STEPS 50' to '#define STEPS 200'.
- The LED blinks when hit contact or win-state is carried out. The fail-state is yet to be performed.
- The 7493 counter still does not count up per mistake contact, causing the 7-segment display to remain at '0'.
- The beeper works as intended, no other misinputs involved will damage the beeper.
- Pre-game warning functions as intended.
 - Combinations of either switch at LOW state will be identified during start-up and will be requested to be turned to HIGH.
 - However, a typical wire game consists in that the hook is powered and the wire is grounded. That can be defied.

See next page for information regarding test number 6.

Breadboard Prototype Test Number 6

List of changes to consider for test number 6:

- Reconsidering the playability and shape of the wire, such ideas include making the wire shaped so that none of its vertical space occupy any of its y-axes more than once in case reverse-motor gameplay is impossible while standing in one area instead of circling around the wire game.
- The code will also be edited so that the beeper is less likely to encounter high-speed signal interference from manual input.

```
142     if (digitalRead(bit_ouch) == LOW && digitalRead(bit0) == HIGH) { // Contact Ouch
143         Serial.println("ZAP");
144         tone(3, 136);
145         digitalWrite(attack, HIGH);
146         cycle = 0;
147         digitalWrite(attack, LOW);
148         while(cycle < 2){
149             digitalWrite(indctr, HIGH);
150             delay(125);
151             digitalWrite(indctr, LOW);
152             delay(25);
153             noTone(3);
154             delay(100);
155             cycle++;
```

- Line 152 was formerly: 'delay(125)'.

But now is edited to be written as lines 152 to 154 from the image above.

- Potentiometer of 10k Ω will be swapped around intermittently with a 1k Ω potentiometer.

See next page for results.

Results for test number 6 are as follows, and noted amends will be sent to test number 7:

- + The 7-segment display works.
- + The beeper buzzes, plus the potentiometer is less likelier to be causing anomalous results.
- + The two switches (inputs) perform as they are intended.
- + The CD4511 functions as intended.
- + Resistors are all flowing their respective currents.
- Changing the potentiometer from 10k Ω to 1k Ω results in that its most resisting option is still audible from the beeper. In 10k Ω , the beeper is instead as close to being inaudible.
- The stepper motor can rotate if the number from '#define STEPS [number]' is not a multiple of 4.
- The 7493 IC is of major concern.
 - Pin 2 was changed from supplying voltage to ground since supplying pin 2 causes the 7493 IC to not increment so it remains at '0'.
 - If pin 2 is linked to a ground rail, the counter will never reset leading to anomalous counting results that are not meant to be carried out. (Decimal 10 to 15 out of hexadecimal's '0' to 'F')
 - What TinkerCAD does not show is that turning the Arduino ON and OFF will result in the 7493 being either persistent or being able to remember its current value regardless.
 - With the current state of the wiring in concern of the 7493 IC, the only domestic way to reset the counter is to unplug and plug the voltage source.
 - And in consequence, the counter increments by 1 every time the game is started or reset.
 - Where even during the modifications of pins 2 and 3 in a power boolean combination to stop the prototype from prematurely incrementing by 1, there must be a way to fix this.

During testing, acquisition of a 74HC93 is null. A 74LS93 takes place.

And to try and rectify the incrementing error, an extra Boolean variable might have to be created.

A chance that the removal of 'pinMode(LED_BUILTIN, OUTPUT);' could suffice and provide an additional physical variable so that the 7493 does not prematurely count.

Breadboard Prototype Test Number 7 – Succeeded Draft Build

List of changes most likely made for test number 7:

- Modification of pin 13 by editing 'pinMode(LED_BUILTIN, OUTPUT);' into 'pinMode(prelock, OUTPUT);'. There, a pin that is assigned to flash the 'L' light by default will bring upon two intentional purposes:
 - 1. Flash when the game is either interrupted by the pre-game command or that the game has reached their win-state or fail-state.
 - 2. Reset the counter to '0' and remain at '0' until the game actually starts.

Results from test number 7:

- The entire board works.
- Ranging from the switches and potentiometer, all up to the ICs and light sources involved. All connections are working as intended.

Since the entire board is proven to work from transferral from TinkerCAD, more tests will have to be conducted to determine how feasible the project will be.

So far, the Arduino program uses under 20% of all available memory.

i.e: 5,578 out of 32,256 bytes (17%) of program storage space and 325 out of 2,048 bytes (15%) of dynamic memory.

Evidence of the breadboard prototype being photographed or code image did not occur during test number 7.

See next page for test number 8.

Breadboard Prototype Test Number 8

List of changes made for test number 8:

- Two additional NAND gates needed for both switches in requirement of changing the switches into makeshift contacts consisting individually of a silver tube bead and a paperclips. It also tests a makeshift hypothesis that the material made from both items can conduct electricity.
However, there is only one more NAND gate vacant in the 7400 on the prototype breadboard so only the hook-wire contact will have to be simulated through the metallic objects as a switch substitute.
 - The paperclips are able to fit through a silver tube bead as well as the breadboard that is used for the prototype. But will they conduct electricity in that the current can flow from the makeshift items to other breadboard components alongside?

Result of the change of switch combination before adding a missing wire that connects from:

- Flawed, but electronic contact appeared to have worked.
- At first, the silver contacts were placed in the wrong location.
But applying a form of current to the makeshift contact produced a result. The output being that the Arduino reset itself fully before recovering.

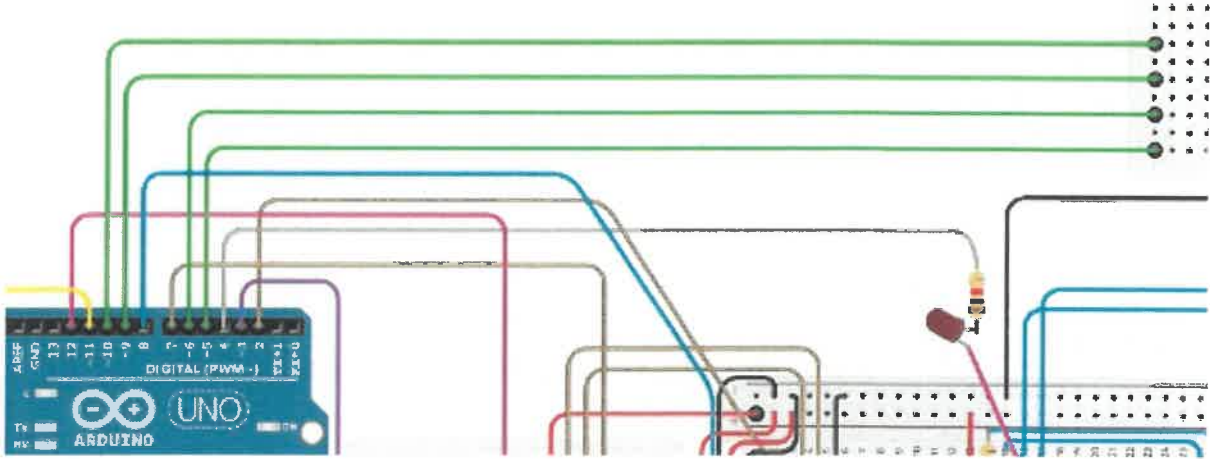
Next objectives after Wednesday, 17 May 2023:

- Reconfigure the switches with a resistor to prioritise current since the rest of the circuitry should have worked.
- And actually connect the power and ground rails since that was forgotten.

Thursday, 18 May 2023

- Explanation why connector pins are implemented on the circuit schematic.
- More prototype tests, including re-wiring to prove whether PWM pins (11, 10, 9, 6, 5, and 3) matter to circuit performance.

Connector Pins on Circuit Schematic and Design



Above: Wiring in which wire connector sockets show an abundance of locations on where they are connected at the TinkerCAD boards – to be edited so to tidy out messy wirework.

Connector sockets and then connector pins provide a less fiddly simplicity to connecting bunches of wires which can be interpreted as having the ability to produce a multi-pin wire connection, dealing less with having to connect individual wires at a time. ✓

Pin Correspondence for Project

Pins will change depending on build type.

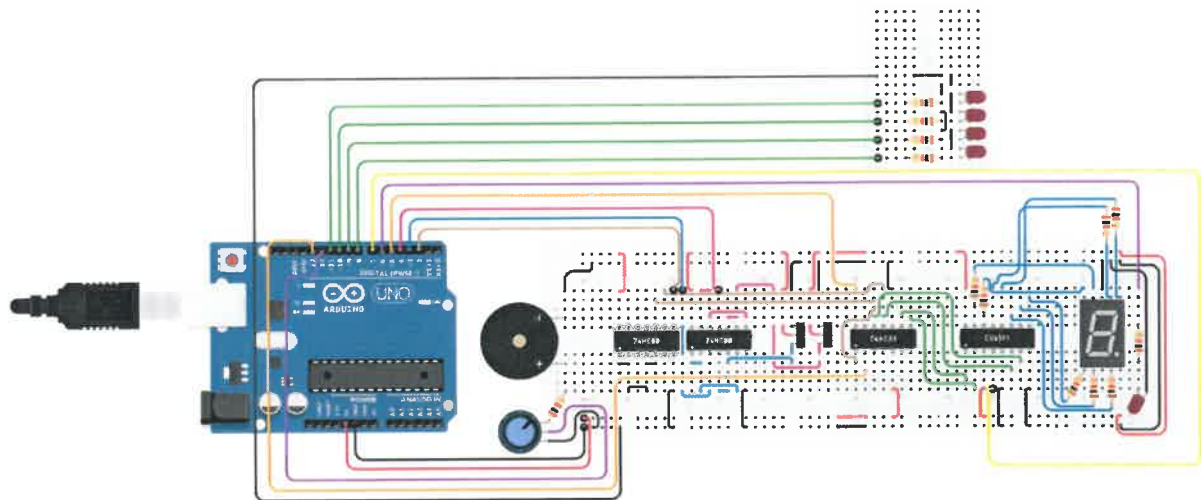
Refer to Project (Appendix 3.4) for table.

↳ you need consistency with your appendix format.

Breadboard Prototype Test Number 9

Almost everything works with the state of the new breadboard prototype layout and code, except that the 7493 does not reset via Arduino's reset button.

It turned out that having to connect both reset pins to the assigned Arduino pin that resets the counter (pin 13) fixed the issue instead of only using one reset pin.



A new physical prototype board will not be in construction since the code from TinkerCAD is semi-compatible with the physical prototype, only that the numbers will be changed for the physical prototype so to compensate for time constraints.

For current working code regarding the TinkerCAD prototype, refer to Project Appendix 5.3.

See next pages for current state of breadboard prototype with reports regarding the tenth and eleventh tests.

Breadboard Prototype Test Numbers 10 and 11

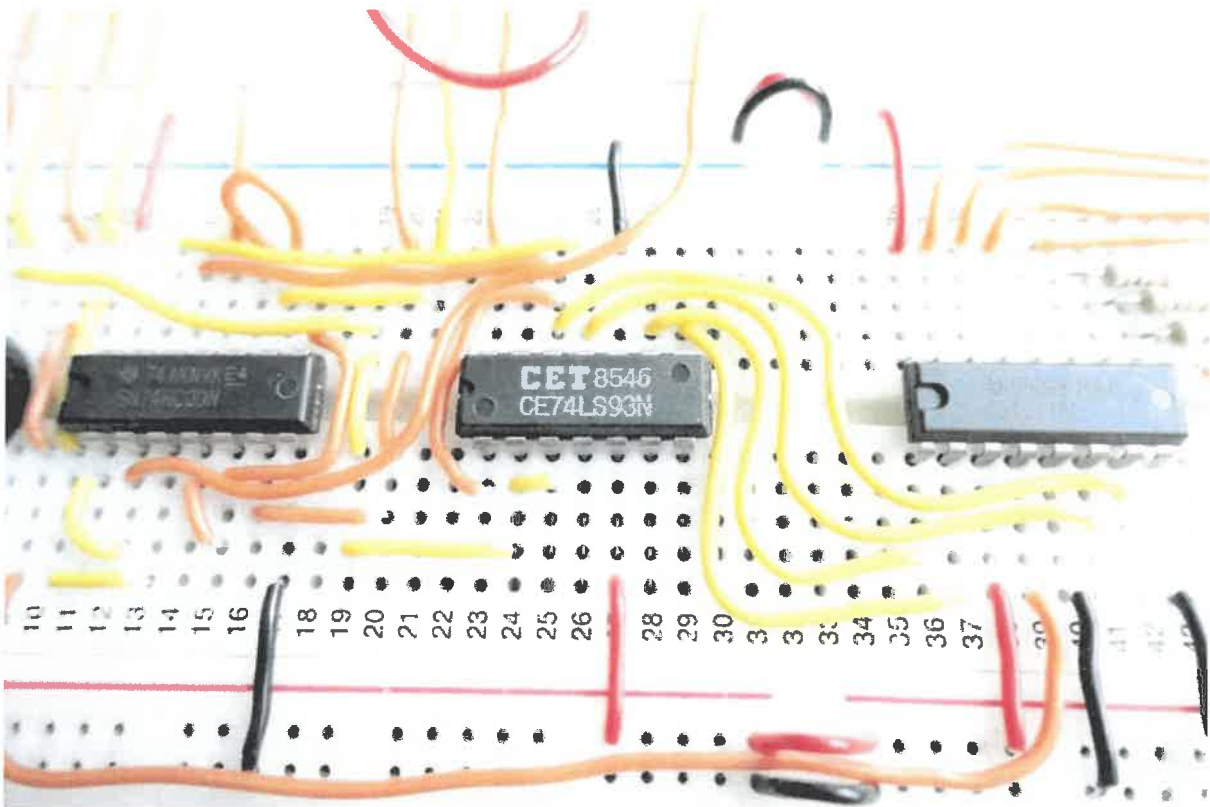
10: Reading contact with switch fails to properly conduct. A resistor will have to bridge across metallic contact and ground rail.

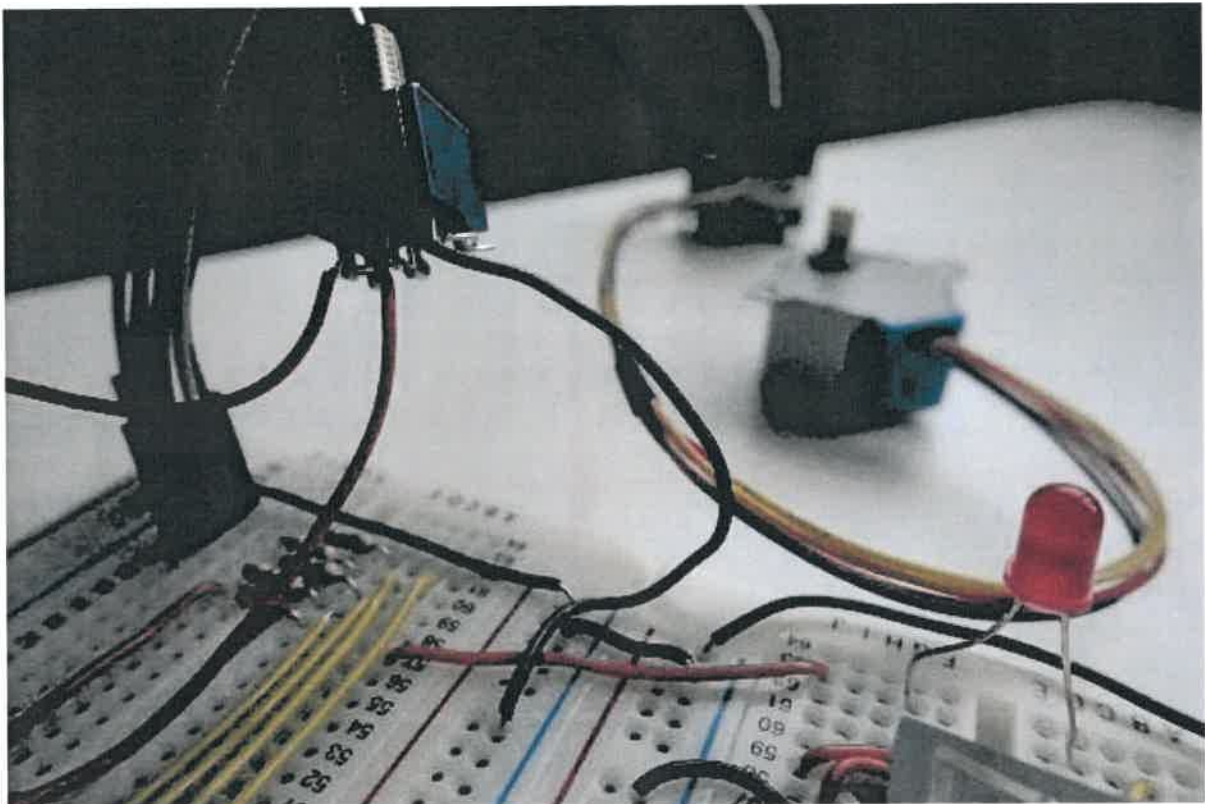
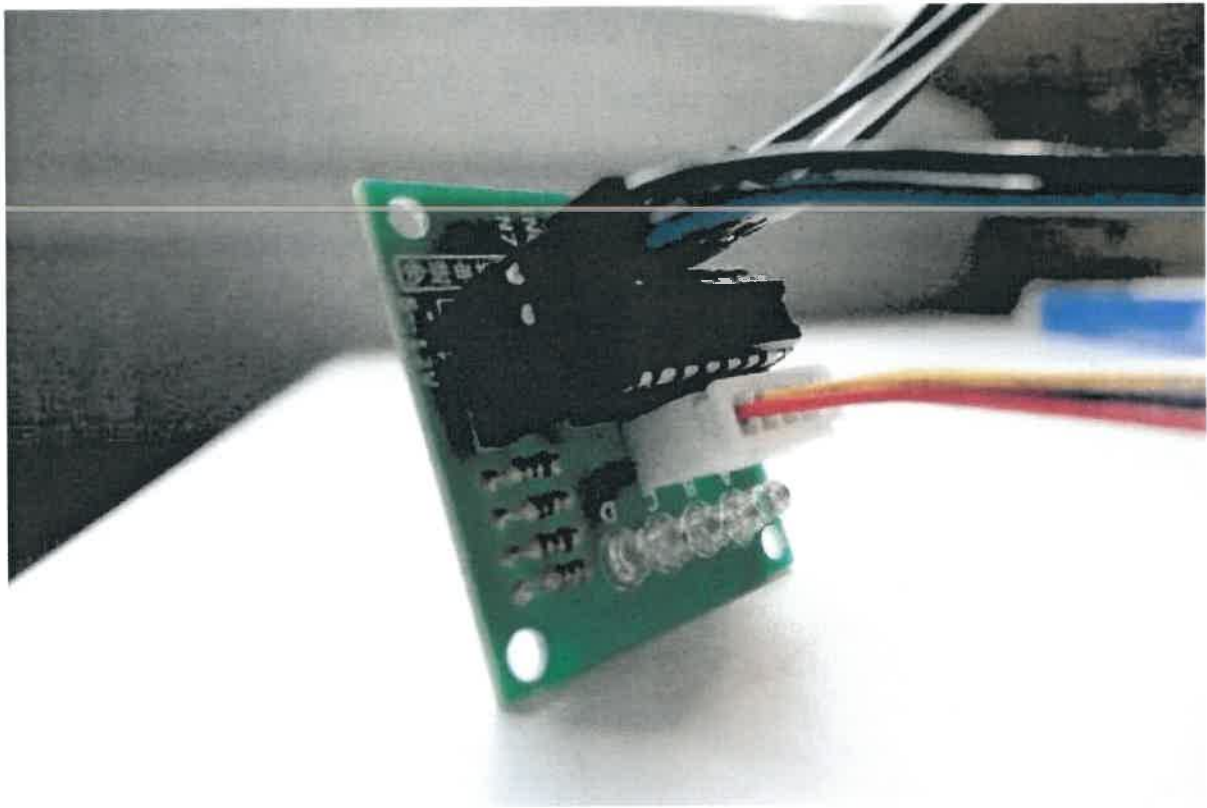
11: Applying a pull-down resistor enables clearer readings for hook-wire contact simulation to properly conduct. Despite the win-state being sensitive on the breadboard, all other functions appear to have carried out as intended.

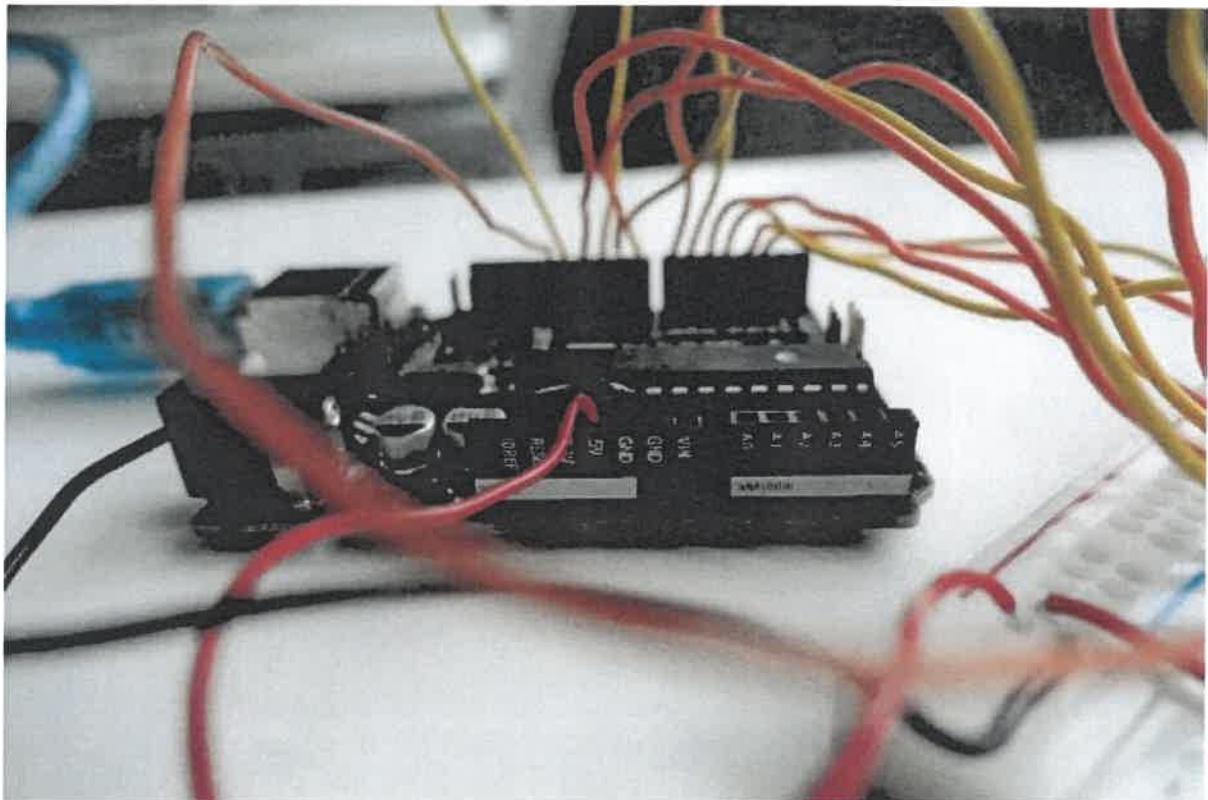
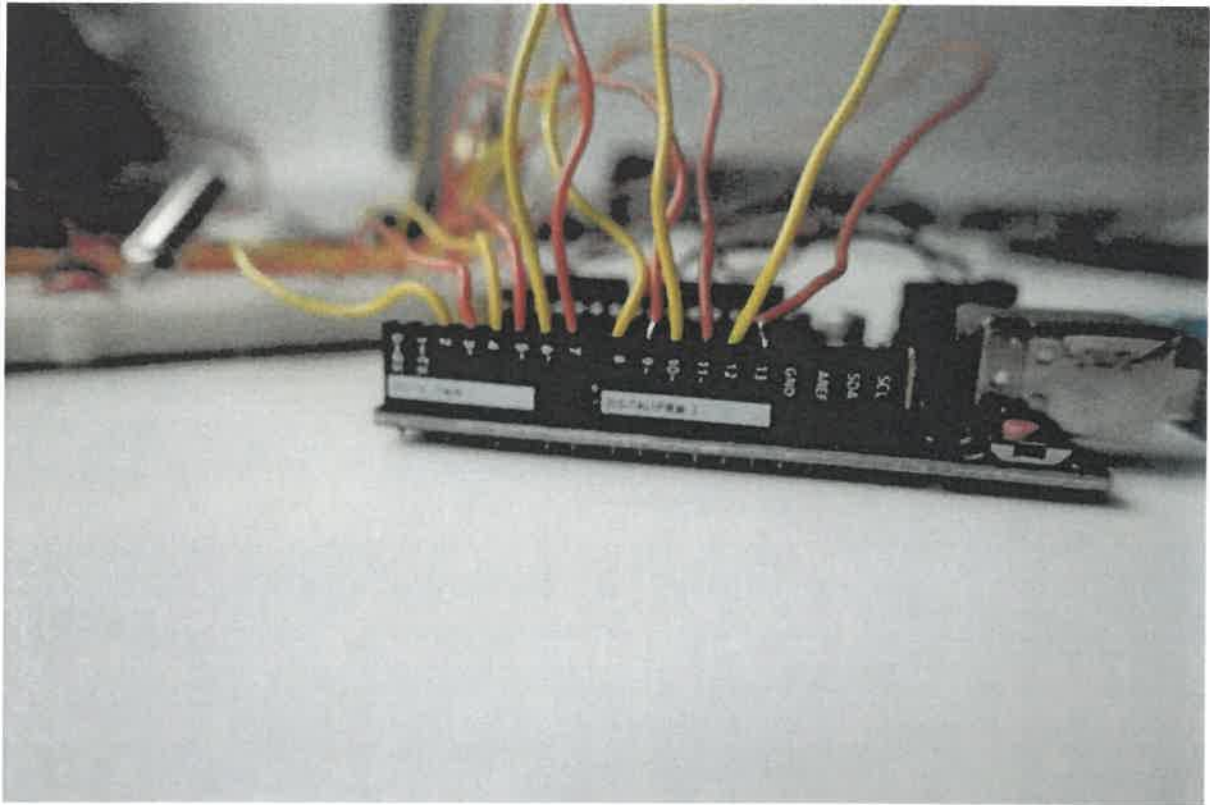
Successful intents across the breadboard! But then to transfer the successfulness into the Fusion 360 software - with influence from the TinkerCAD rendition.

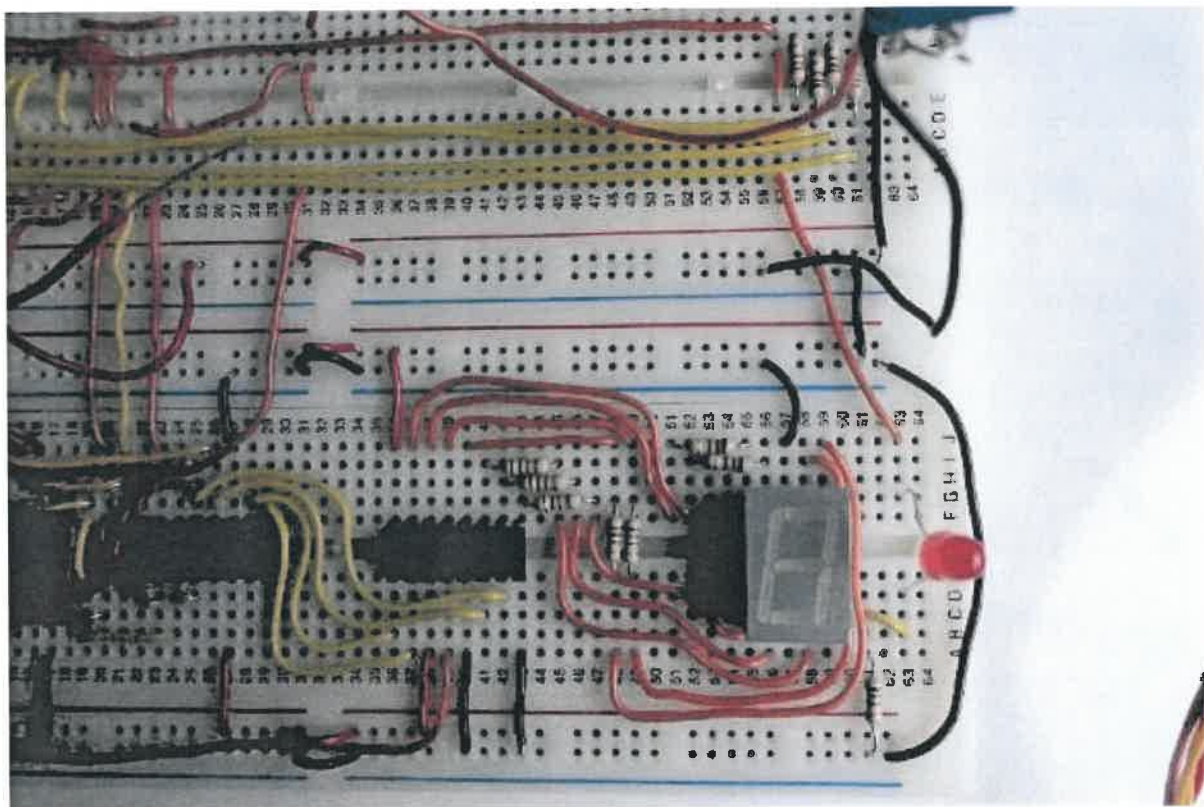
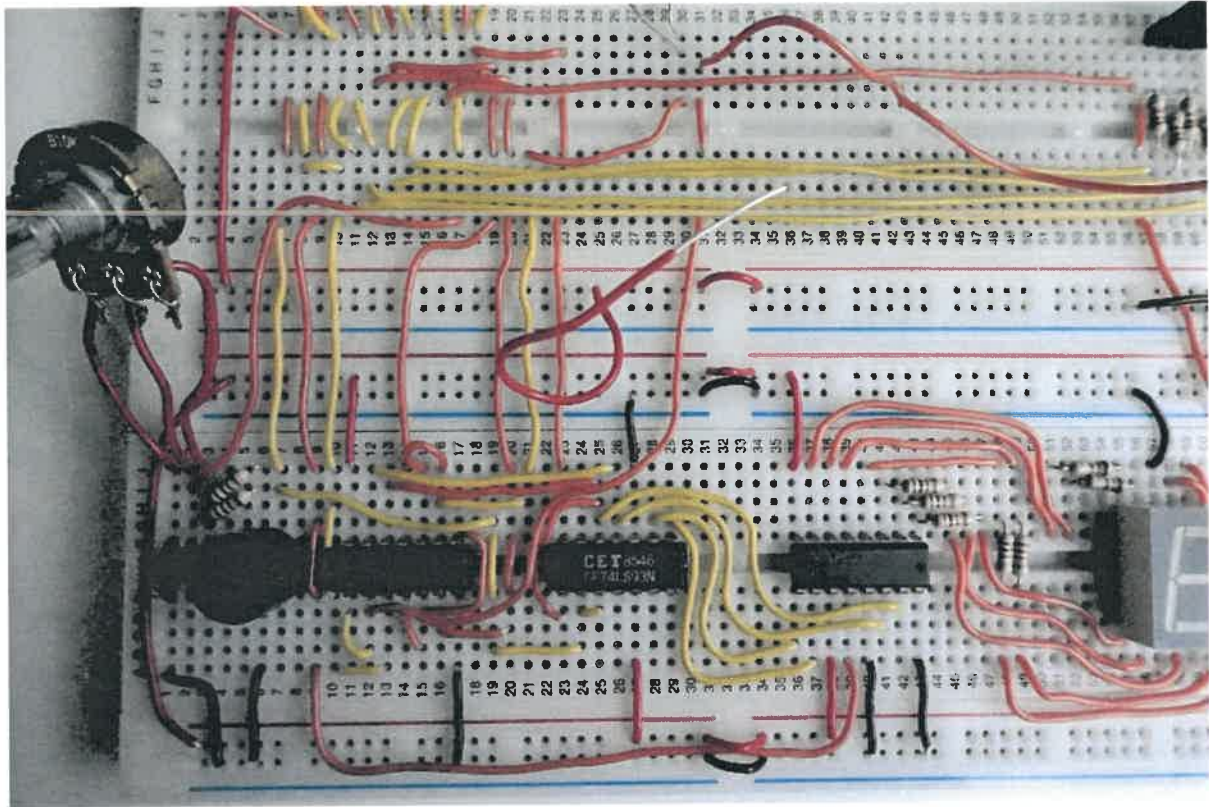
What should be left afterwards is converting the remaining toggle switch (win-state trigger) into another metallic contact, all as long as neither the win-state trigger nor the 'wire' touch each other.

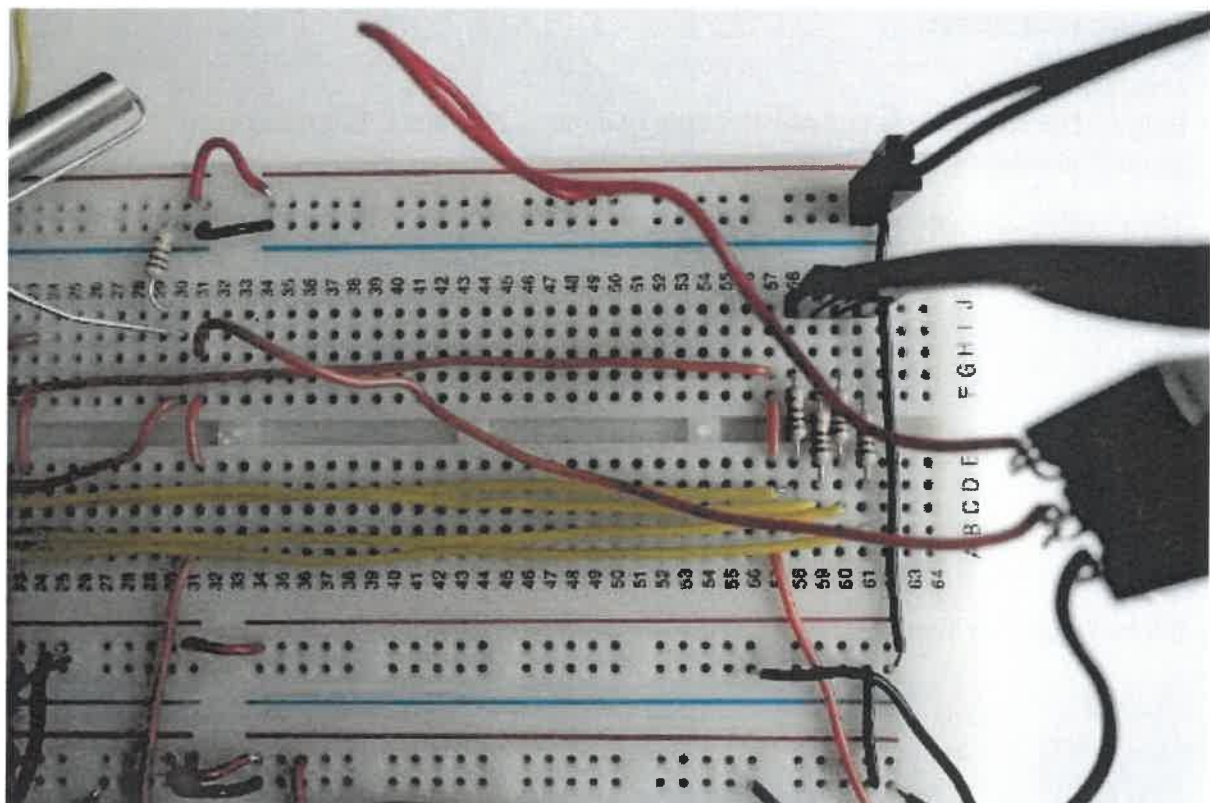
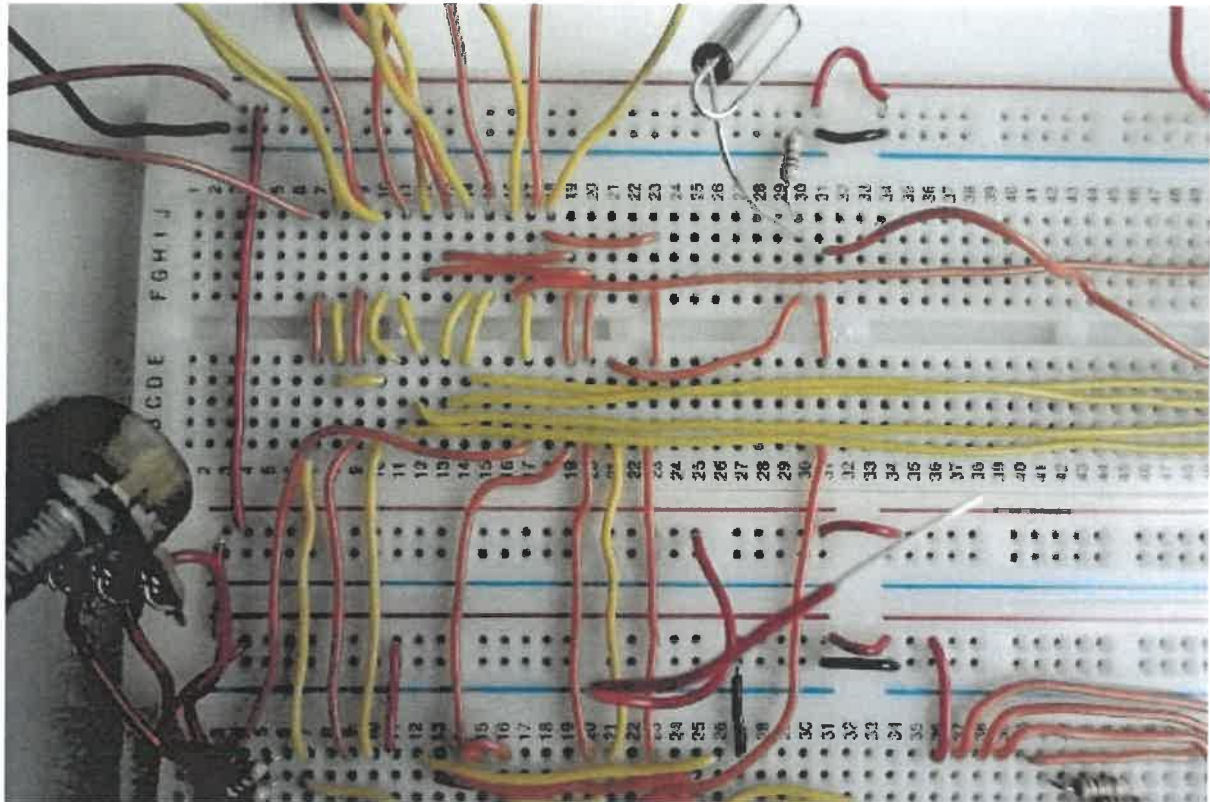
Images of the successful build will be displayed here and through the next few pages.

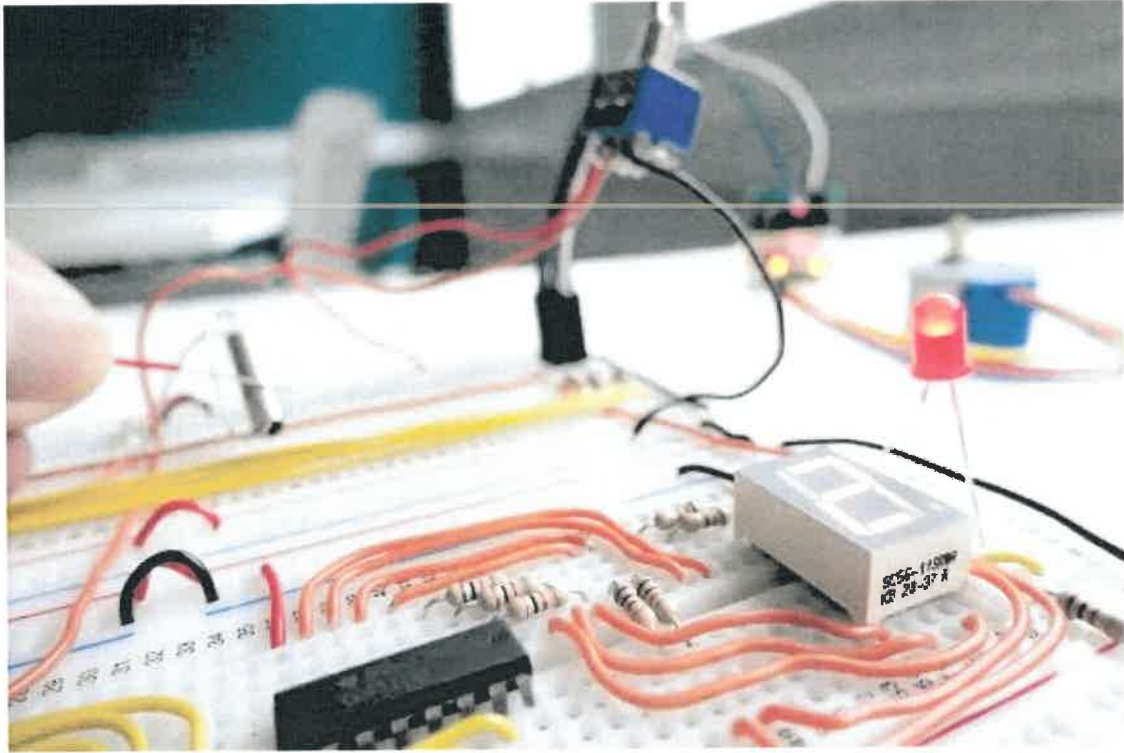










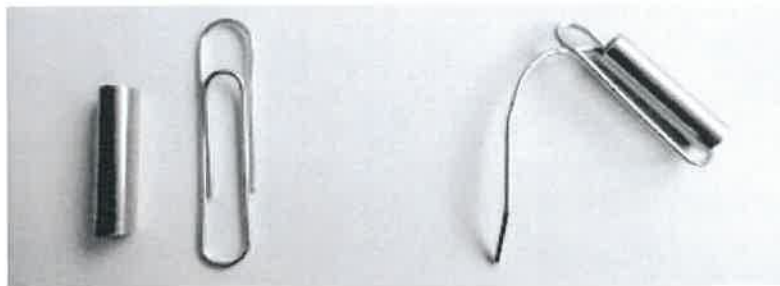


Above: Testing the contact with a +5V wire of the same current flow from the Arduino and programming device.

Metallic Contacts

Below: The components and one iteration of a metallic contact. Proof that their materials can conduct electricity.

The metallic contacts will be manually produced for a reasonable quantity so as to fit for the final product.

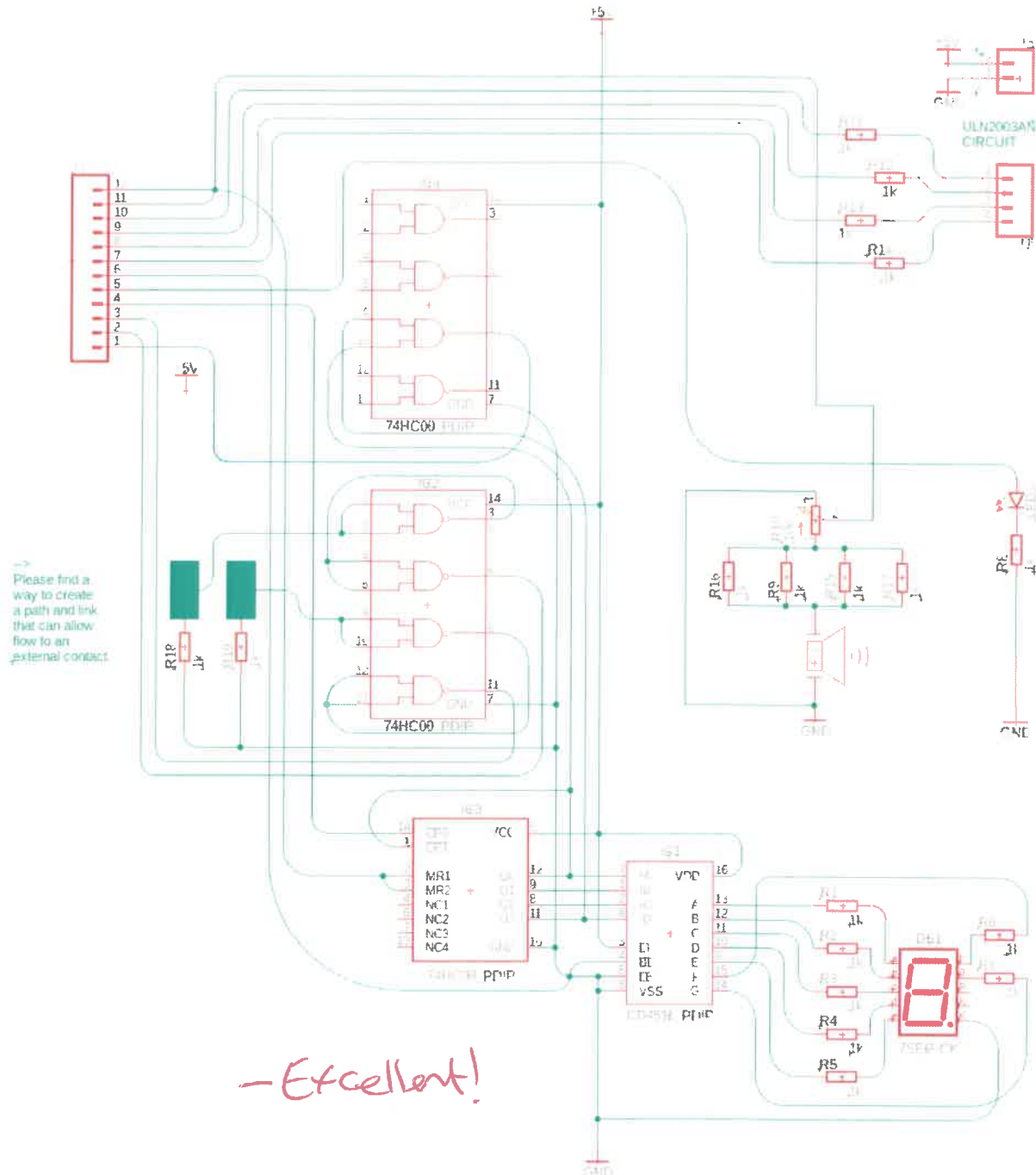


- Excellent!

See next page for first iteration of manually-produced batch.



Fusion 360 Current State of Circuit



Meanwhile in Fusion 360, as the coding is partially finished, circuit schematic is close to being finished. The current obstacle lies over what the metallic contacts should be connected as.

In turn, the thought process to tackle the issue takes significant foresight and to generate a plausible set of answers.

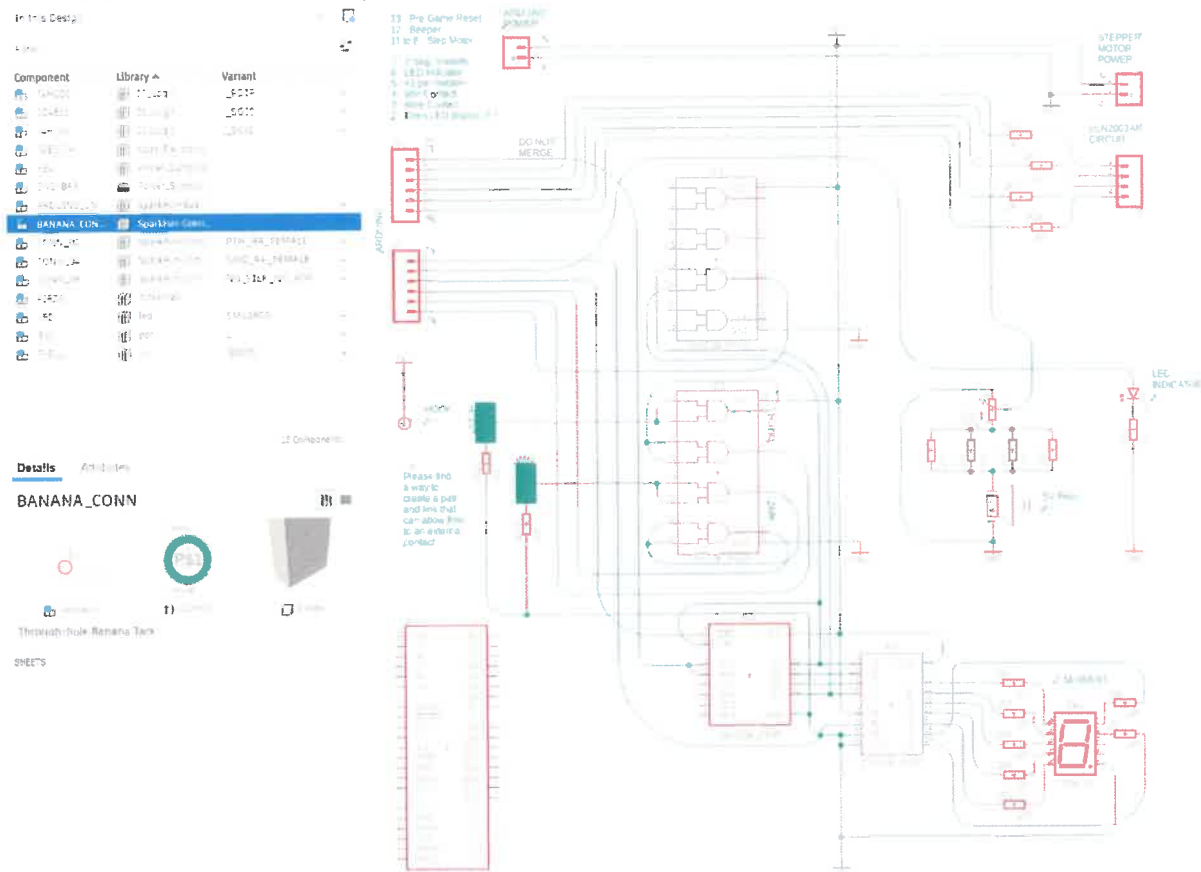
- Good reflection.

One type of connector that could be applicable to the circuit is a 'banana' connector. They can be found with older machinery such as DC voltage supplies.

Monday, 22 May 2023

- More states of Fusion 360 circuit schematics.
- Circuit Design begins, with having to swap out components to find correct sizes.
- Visual aiding comparison by placing components on the screen that shows Fusion 360.
- Development with components such as:
 - Wire Stand
 - Wire Stand Holder

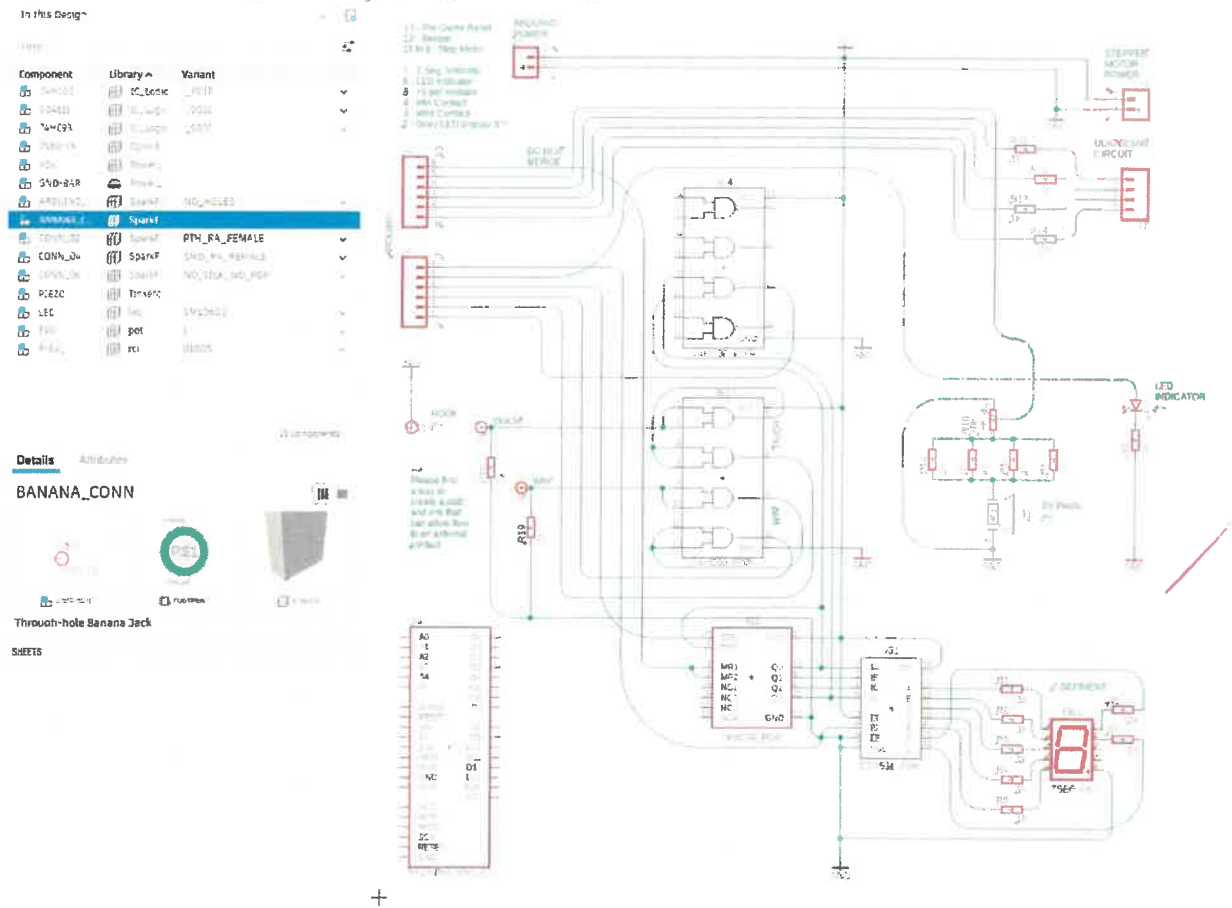
Circuit Schematic of 22 May 2023, Iteration 1



Pins 4 and 3 (Win Contact and Wire Contact) are still up for individual debate for deciding what component to elect for external connection to let current flow through rectangle-represented areas.

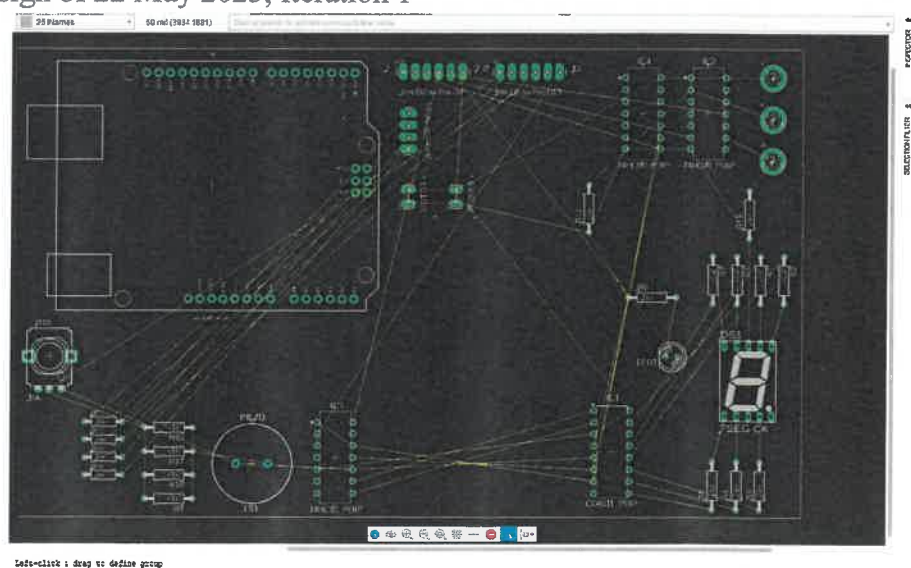
See next page for next iteration.

Circuit Schematic of 22 May 2023, Iteration 2



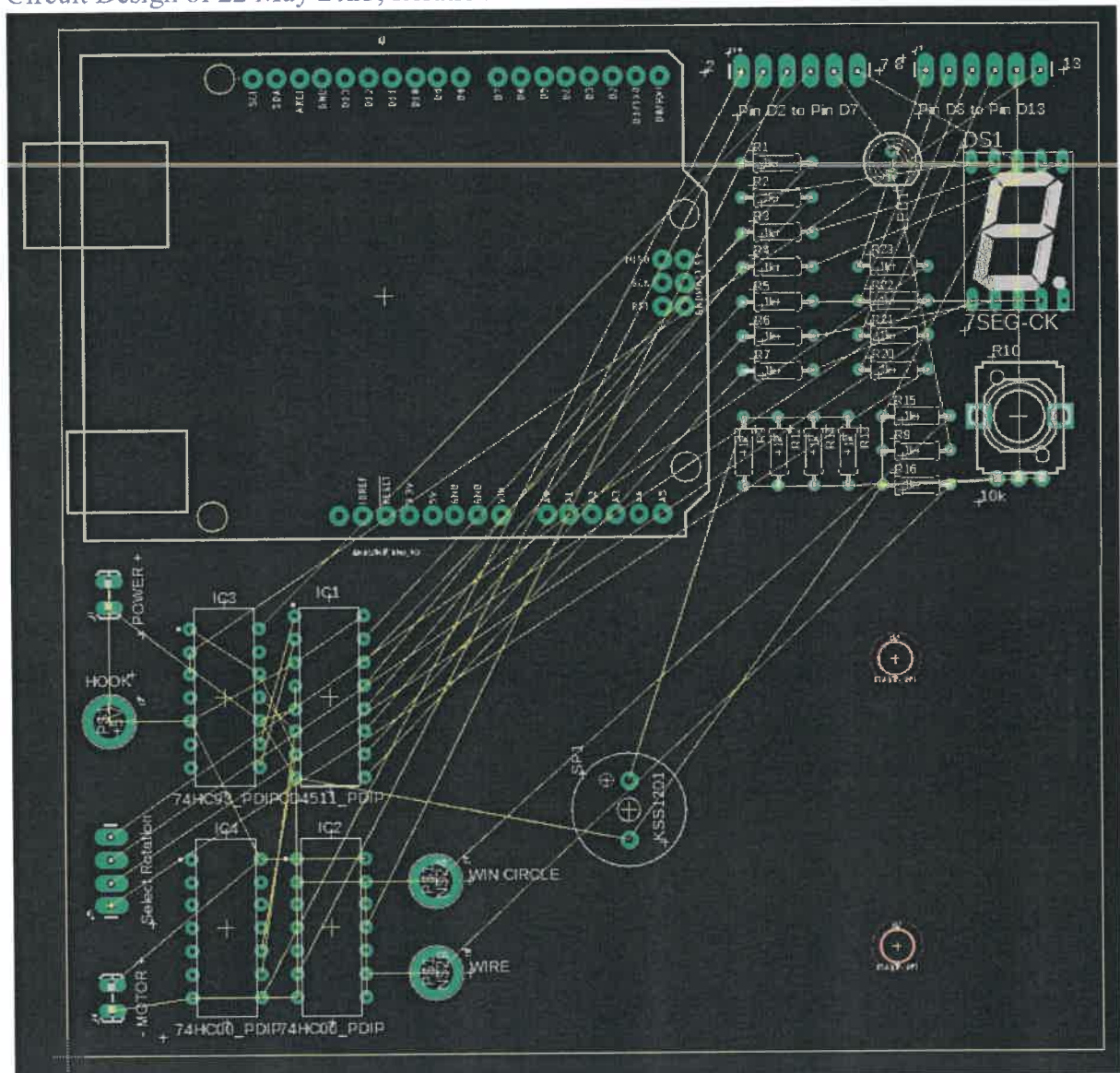
For unspecified shortcut reasons, more banana connectors have to be implemented so to act as circuit recipient substitution points instead of testing points that are like hooks.

Circuit Design of 22 May 2023, Iteration 1



Once entering 'circuit design', the need to replace components with different names and variants in the schematic will be carried out alternatively with component placement in design in case perceived measurements per component are not what as they are ideal in intended design.

Circuit Design of 22 May 2023, Iteration 2



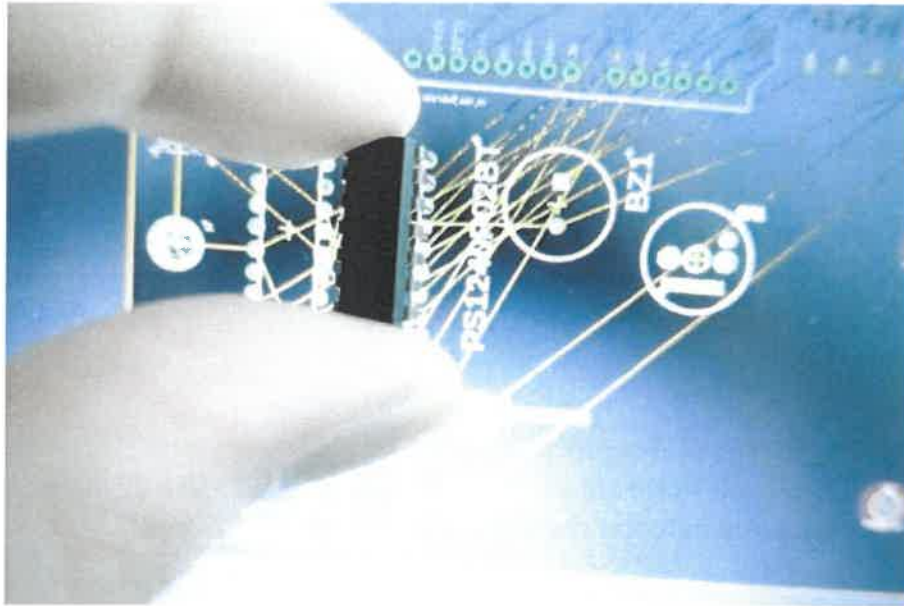
One of the components missing from this iteration is the stepper motor.

The stepper motor might have to be interpreted as screw holes instead since 28BYJ-48 does not exist in any of the stock Fusion 360 component libraries.

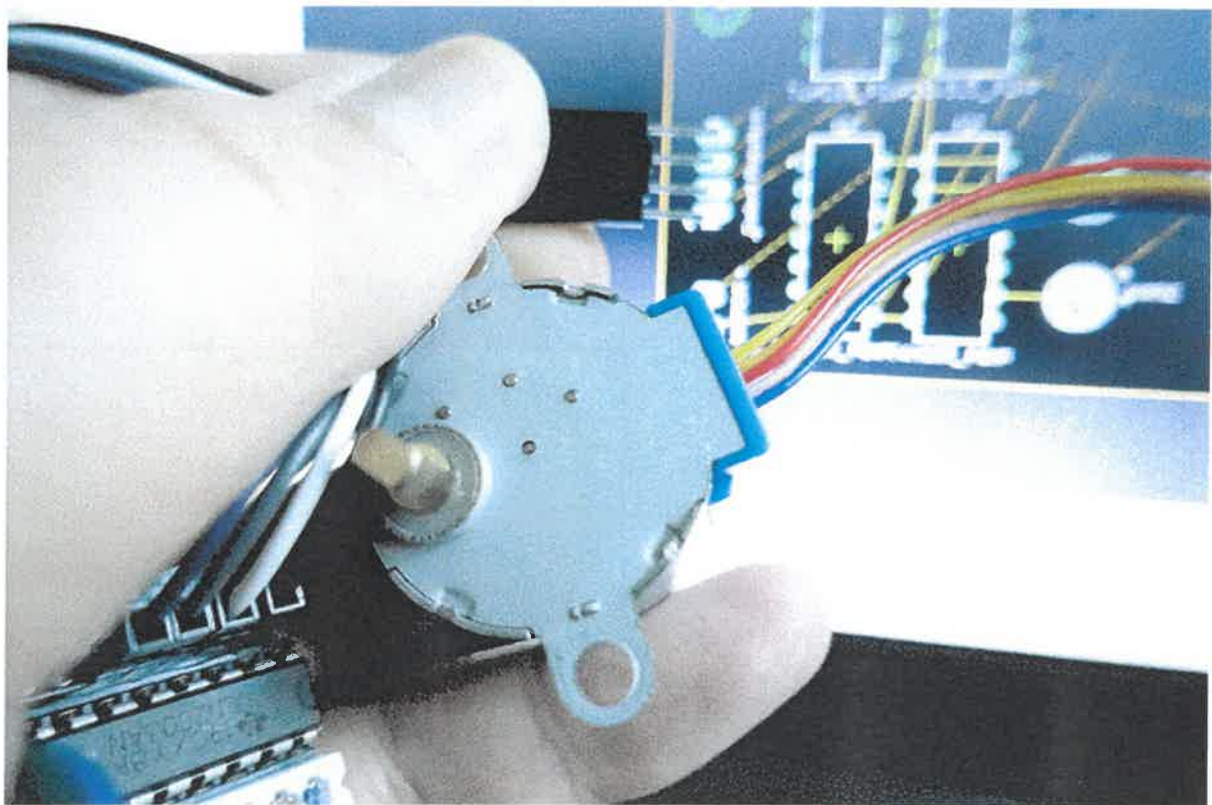
See next page for need of aided component size equivalence for perceptual references.

Finding Comparability with Real Life Components

For comparison in assurance of component sizes, a CD4511 is placed on its allocation.



Another example sets the four male-to-female pins connected to a 28BYJ-48 circuit in comparison to the screen.



Hook Design, Iteration 3

Due to intuition-based measurements and how handling of the hook design is to be regarded as per user considerations, a new hook design is produced.

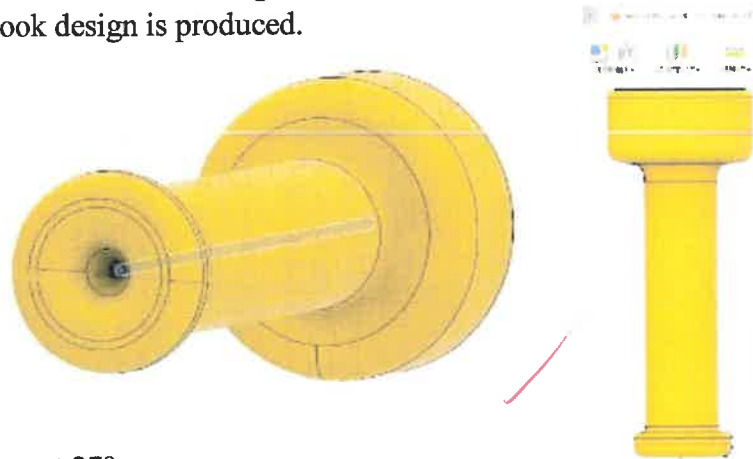
Height: 15cm.

Inner Radii: 1cm

Outermost Radii: 30cm

2nd Outermost Radii: 20cm

Handle Radii: 15cm

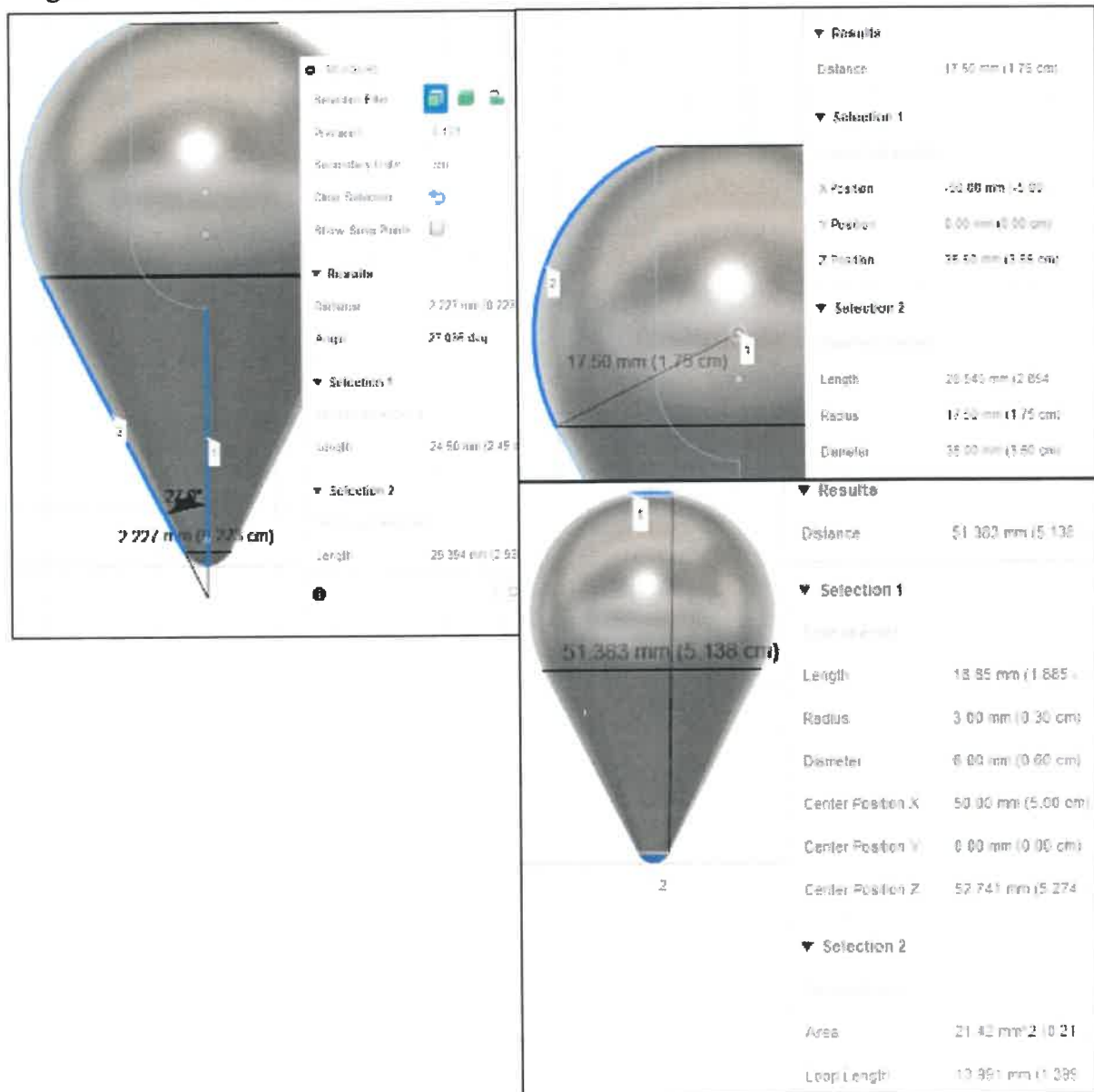


Wire Stand Design, Log 2

The angle of the wire stands measure at 27°.

And the widest the stands extend in a horizontal axis is around 35cm diameter.

Heights differ between stands, but the tallest estimate is around 52cm.



Wire Stand Holders – Log 1

After three iterations (from left-to-right order), it was intended that the angle of the wire stand holder should be slightly wider than the wire stands (27.036°).

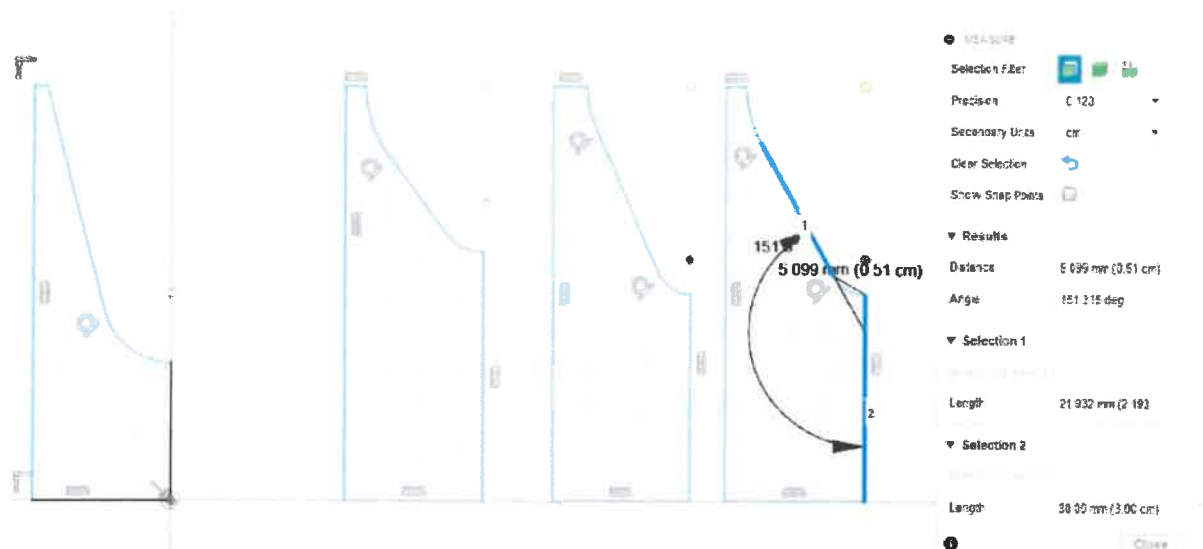
It is intentional in being measured in angle's parameter due to concerns regarding an imbalanced wire with the inability to maintain friction around the wire's rolling holders (which is to be designed outside Fusion 360; paperclip width is 1mm).

Iteration 1: 14.5°

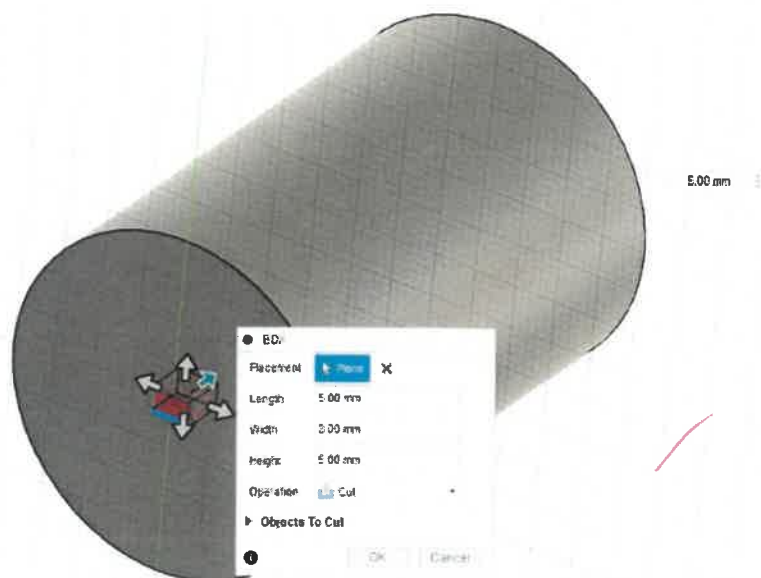
Iteration 2: 36.0° (-144.0°)

Iteration 3: 23.5° (-156.4°)

Iteration 4: 28.6° (-151.3°)



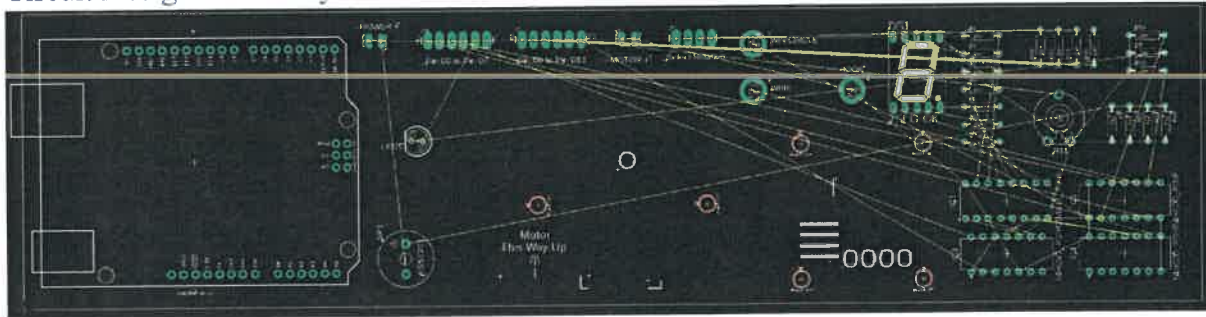
As for producing a notch for the wire stand, manual measurement using a ruler on the physical 28BYJ-48 stepper motor component helped require measurements starting from the bottom of the wire stand holder to stretch it up to the measured height of the stepper motor pivot.



Tuesday, 23 May 2023

- Continuation with Circuit Design

Circuit Design of 23 May 2023

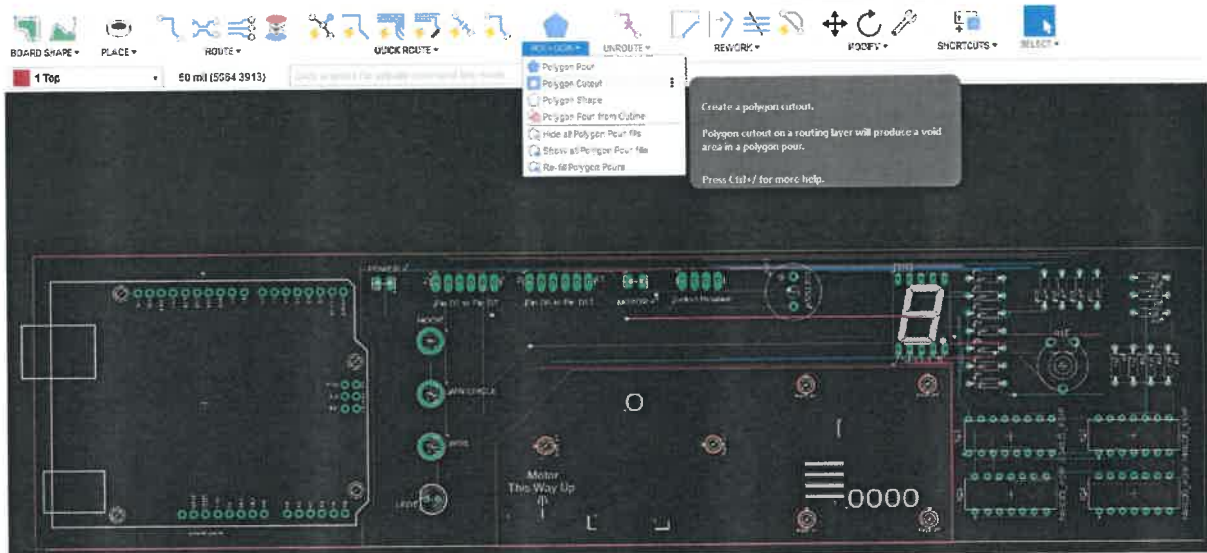
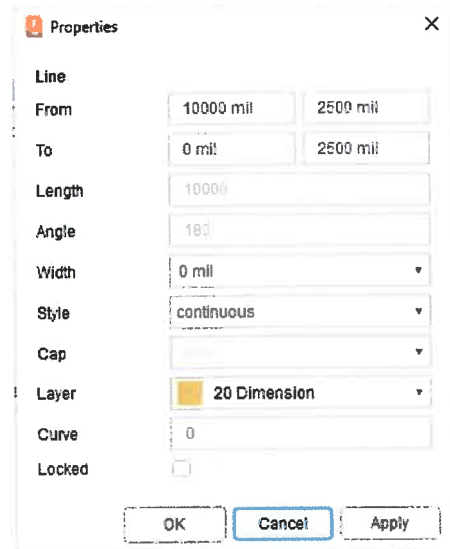


The next state of the circuit design is modified so that it prioritises towards less use of circuit board material (from approx. 5500 mil in both dimensions) and to fit the circuit boards used for the project.

The next step is to apply areas where copper must not print in the areas where other circuit boards should be located.

'Polygon Cutout' was used so that 'Autorouter' (last or 6th option above the 'Quick Route' drop-down) does not touch the area of the circuit where external circuit boards might be affixed to.

Fortunately, the routes appear thinner than was subjectively anticipated given that routes on older circuits appear around 2 millimetres thick.



Pushing the Circuit Design into 3D PCB

Refer to Project Appendix with corresponding chapters below:

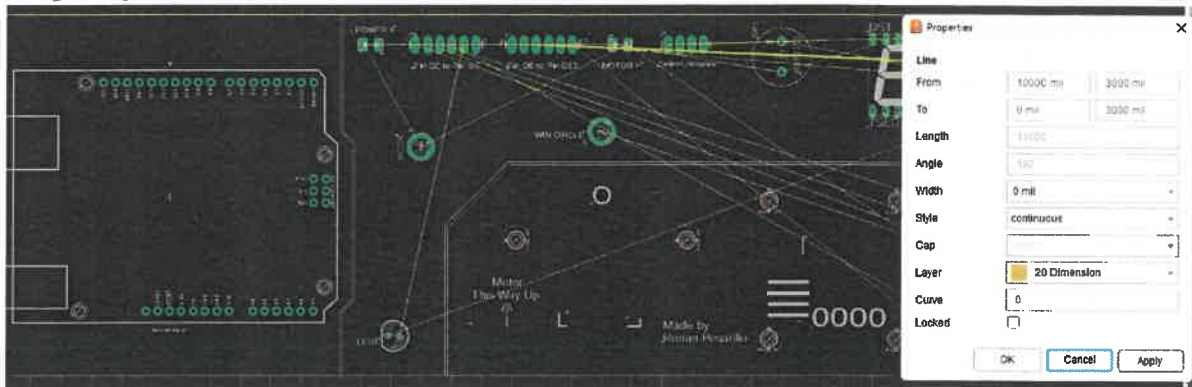
1.1.7 for pushing a circuit design into a 3D simulated PCB.

1.1.8 for placing text on a circuit design.

Wednesday, 24 May 2023

- Giving the PCB 500 mils wider in width and the impacts it brings.
- Does inserting derives produce components for the entire project?

Temporary Circuit Board Extension from 2500 mil to 3000 mil



The decision to extend the PCB is for purposes of considering how much will the external wires involved in the game will endure before they snap or disconnect internally. Conversely it can be considered a “waste of space/material” if development convenience were to go by, especially if decisions to produce the casing for the game is to be the next step.

Eventually, dimensions are kept to 10,000 mil × 2,500 mil.

Side Note: Watch carefully with whether component pins are assigned to unintentional nets. As an example, pin 7 of the female pin connectors that link to the Arduino UNO might be assigned automatically as a ‘GND’ pin, so editing the schematic must be carried out before further rendering is involved.

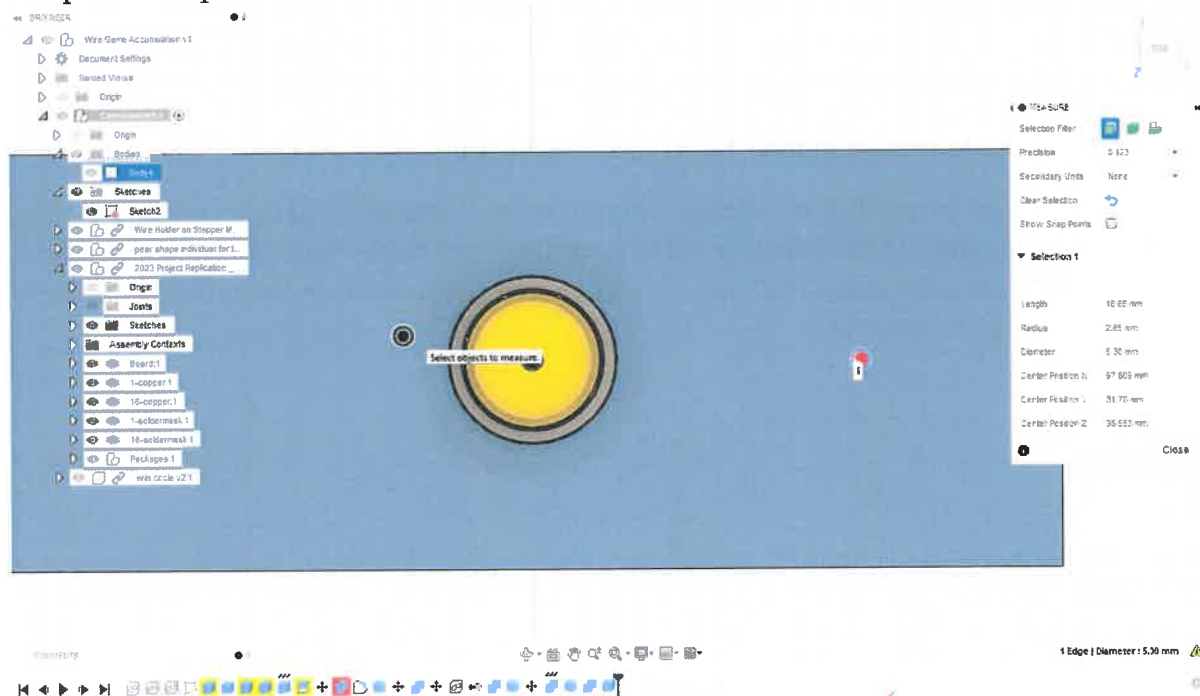
Inserting Components to a New Project

Refer to Project Appendix 1.1.9 for instructions.

Beginning of Assembly and Physical Modifications



Inserting components to a new project will mean that there will have to be navigation through component drop-down as seen on the left of the above image.

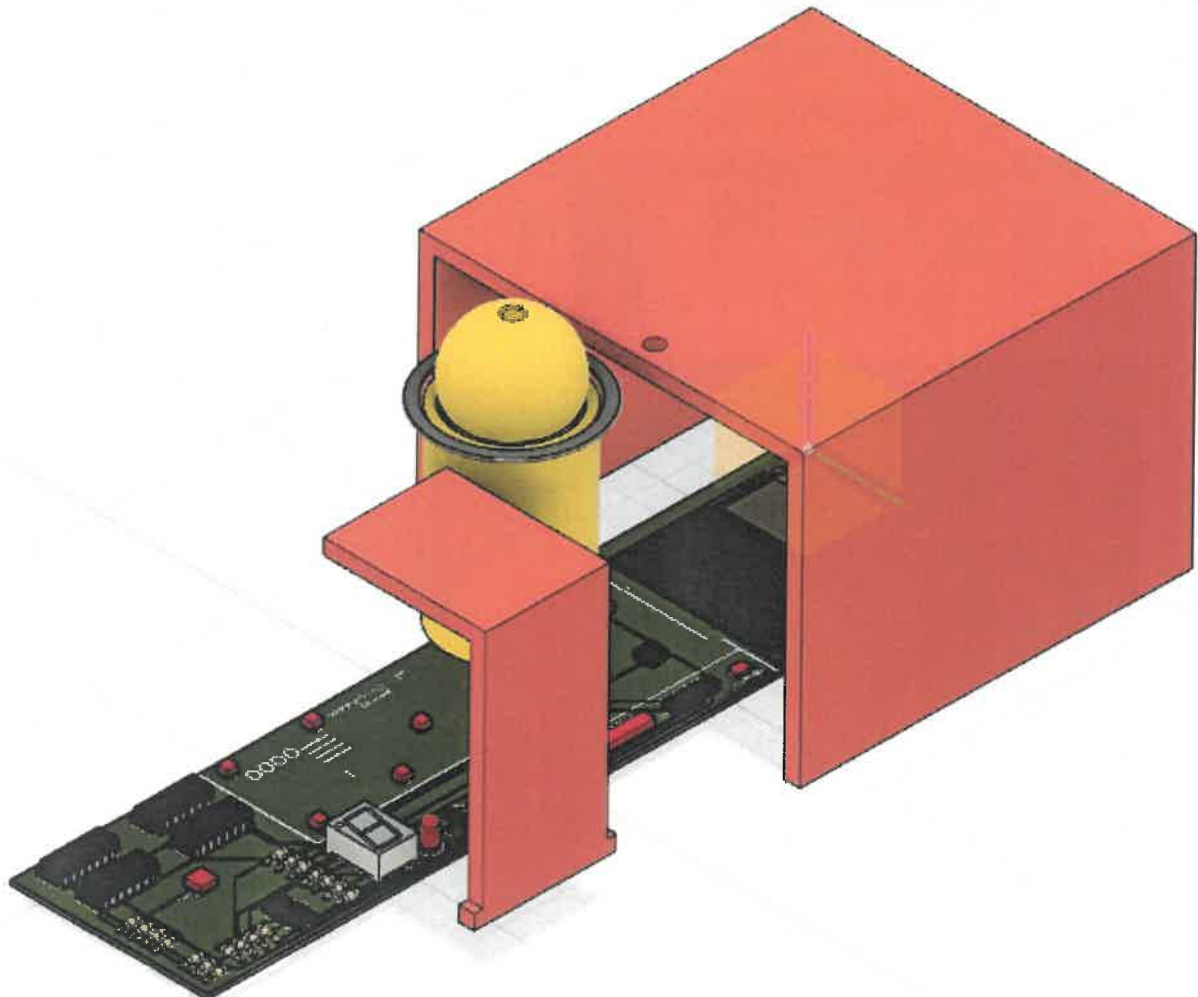
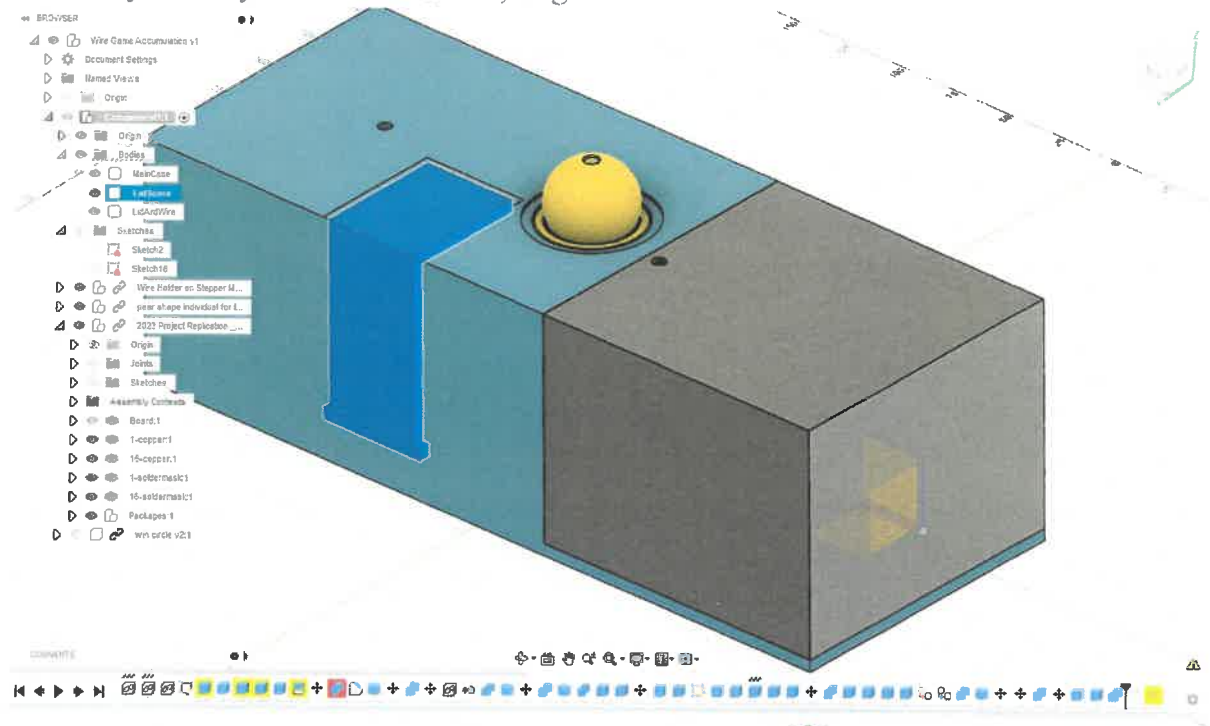


Decals for the project's box are not to be implemented yet.

Currently, splitting of the project's box should be reminiscent of removing the lid of a remote control's battery compartment.

Components such as a hook (although designed individually) and an iron holding rod with the grapple made from metallic contacts to hold the spinning wire from dropping are yet to be demonstrated.

Assembly and Physical Modifications, Log 2



Thursday, 25 May 2023

- Marble and Metallic Contact Makeshift Visualisation on Fusion 360
- 3D Printer Unexpectedly Miniscule
- State of 3D Model
- 3D Print Preparation

Marble and Metallic Contact Visualisation

Refer to Project Appendix 4.3 for chapter contents.

Making Do With A Household-Sized 3D Printer via. Jigsaw Cut Shaping

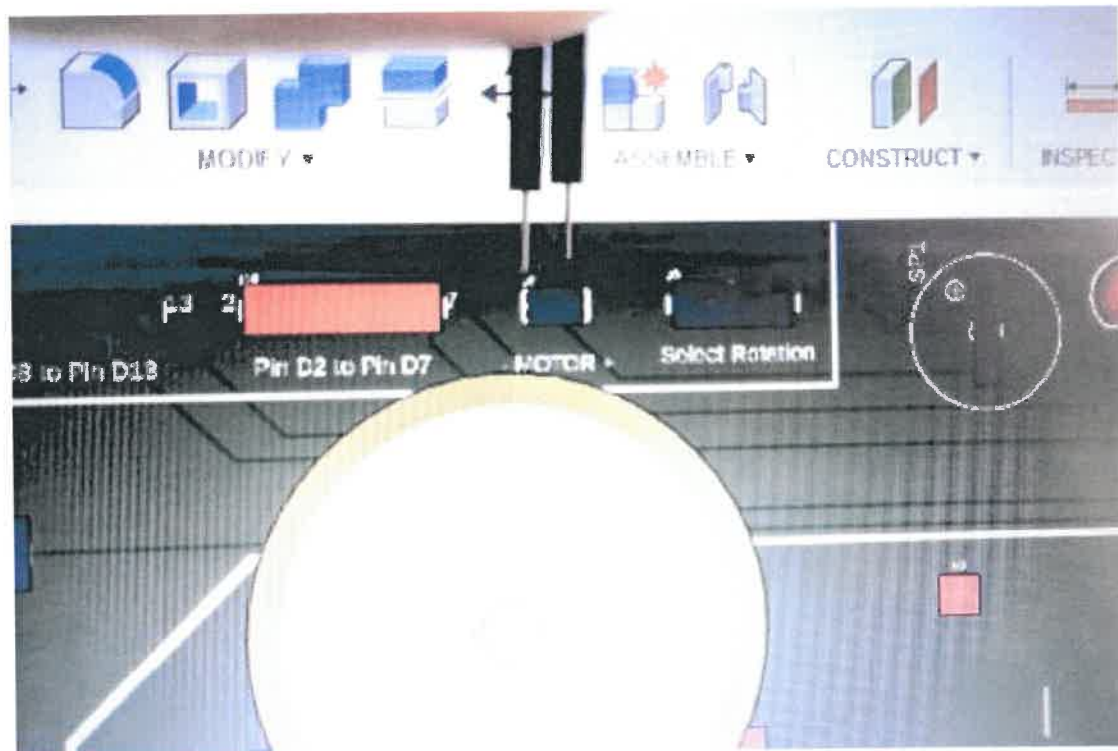
Considering that the dimensions of the 3D Printer being used for the project end up being inadequately small, the idea of splitting the game base into pieces less than 15cm in all three dimensions will be carried out; where 13.5cm might be for affirmation purposes.

After consultation with machinery parameters, they confirm dimensions to build is '150mm x 150mm x 150mm'. The machinery also supports USB and Wi-Fi communications.

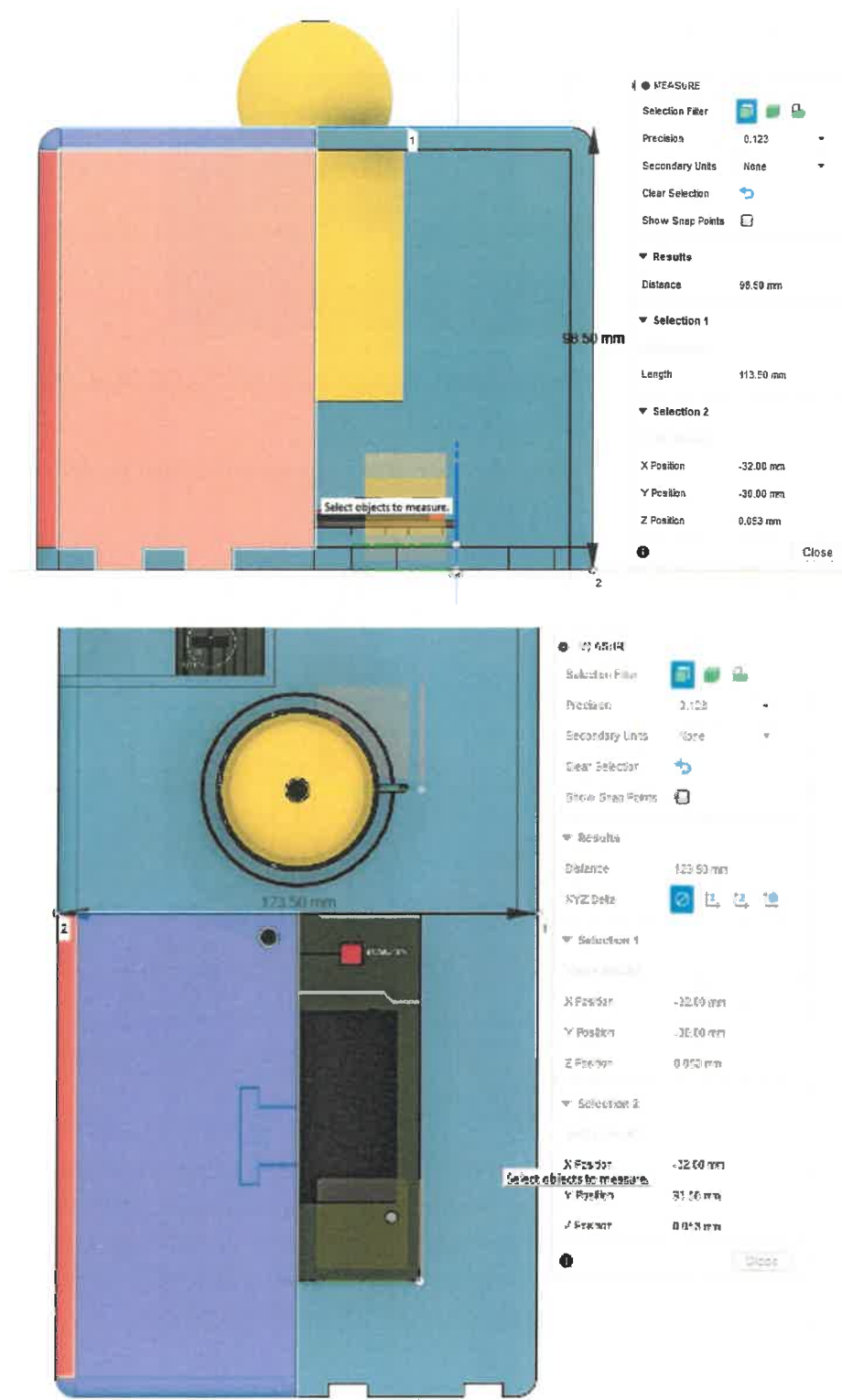
State of 3D Model

Currently where evident, the pin connectors cables for the 28BYJ-48 circuit can barely fit inside the plastic case and around the component that is spun by the stepper motor component. Fortunately, the plastic casing is wide enough so that large amounts of cable can fit and inflict less bending damage.

See below image for bare proof of fitting.



See next page for current parameters of wire game.

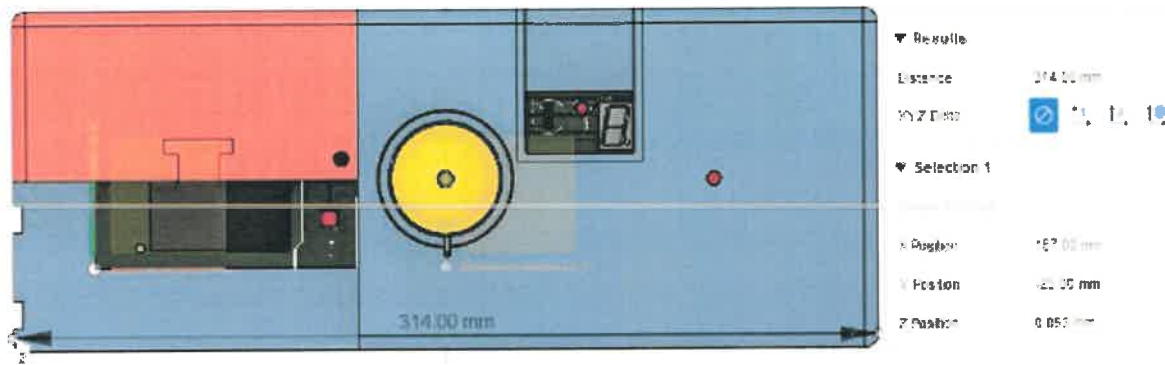


Maximum Estimated Dimensions:

L: 314.0 mm

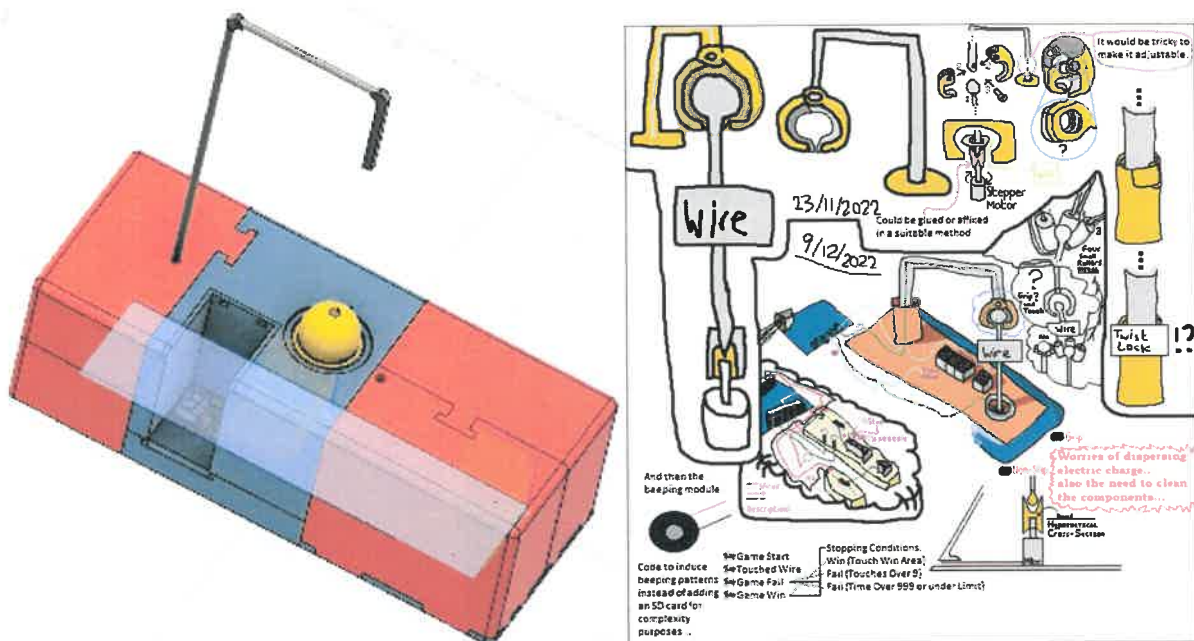
W: 123.5 mm

H: 98.5 mm (+ 30.0 mm from wire holder = 128.5 mm)



As stated, the 3D printer's maximum dimension of printing is 150mm in all height, width and length.

Considering that the length is 314mm, the wire game casing/base must be split into no less than three sections.



In the event of producing a prototype on Fusion 360, time constraints can cause the simulated product to look simpler than is desired. Fortunately, the illustration months before did promise that the product would be written with realistic prior knowledge regarding the physics, current flows and object friction recurrence. Albeit a few other details presented that would not immediately qualify with said cause of rushed development.

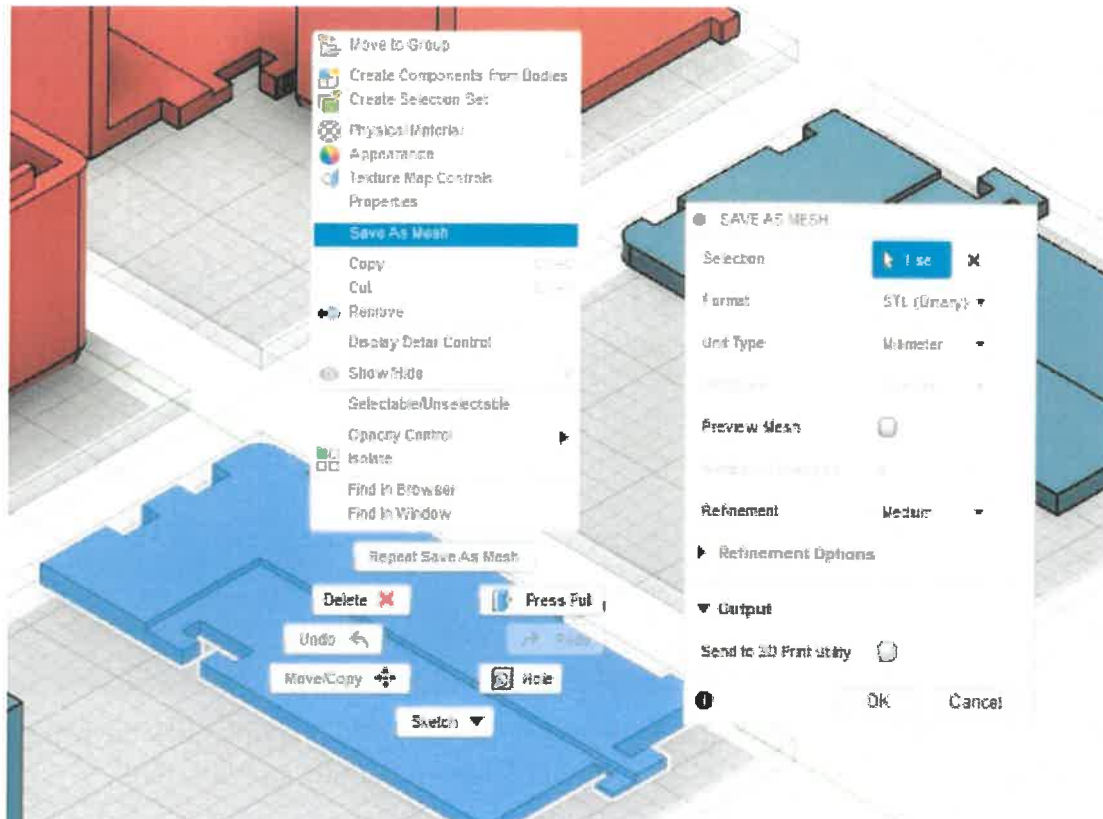
Further reflection/analysis could be added here.

Rather than focusing on what happened - how did you overcome it and progress?

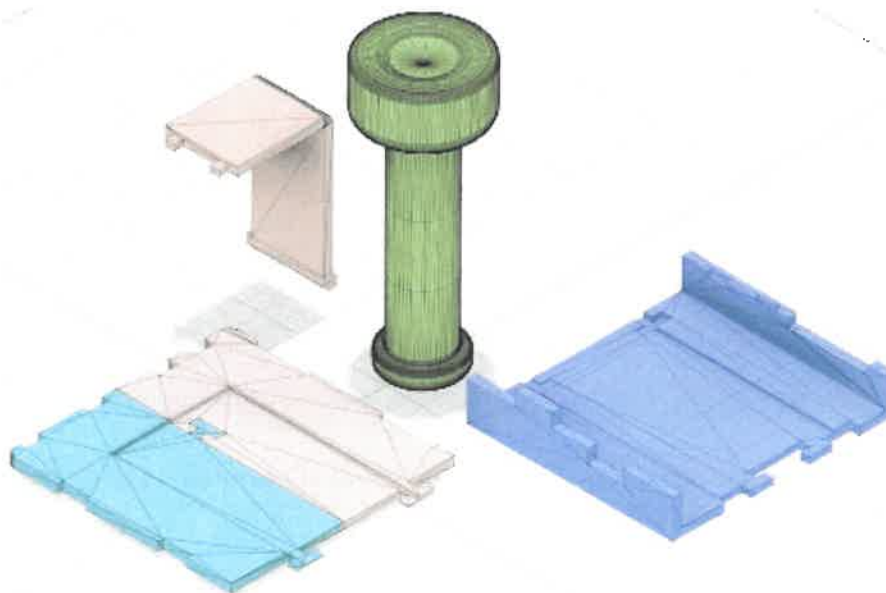
Preparation for 3D Printing

After designing plastic pieces for the wire game, they are then arranged via.: 150mm or 170mm square surfaces laying horizontally; clicking the ‘Modify’ drop-down and selecting ‘Arrange’; selecting all relevant components to print while on the ‘Objects’ tab; allocating them to as few square surfaces possible while on the ‘Envelope’ tab; and rotating them where respective concerns are quickly handled.

Right-clicking each part and select ‘Save As Mesh’ (or previously ‘Save As STL’).



As each component are now .stl files, they can be prepared for 3D printing unless further modifications are to be made.



Friday, 26 May 2023

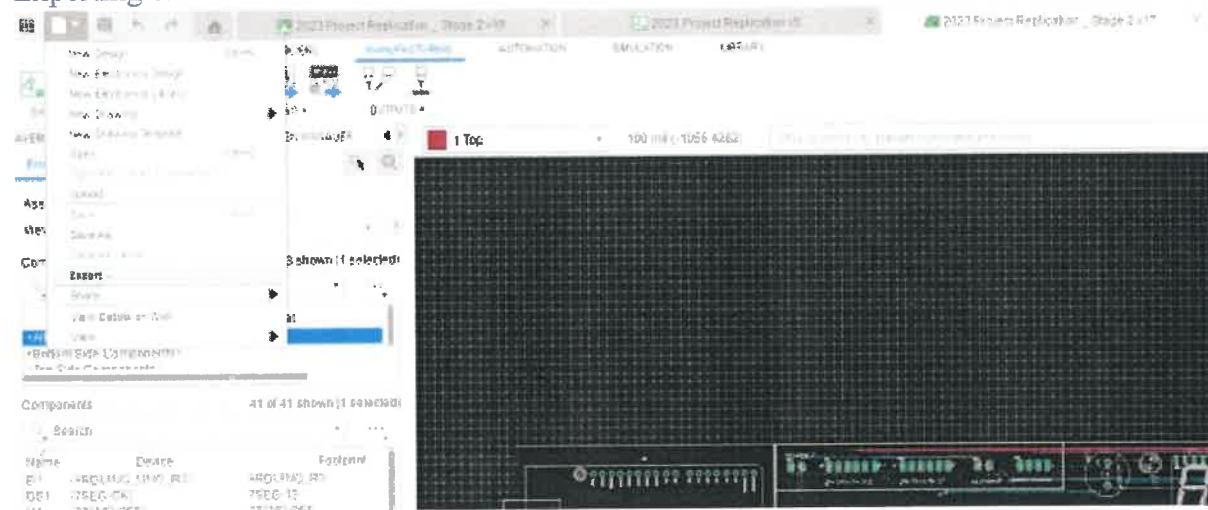
- Before Mesh Printing Amendment
- Exporting of PCB
- Proof of Printed Model(s)
- Experimentation with More Metallic Products such as a Sink Strainer
- Makeshift Arrangement of Props

Before Mesh Printing

As stated in the previous page, .stl files might have to be modified in case of any difficulties arising. And it already happened with the hook as it was almost 15cm in height.

10mm of its height, in estimate, is removed to compensate.

Exporting of PCB



As the PCB has not been reported to be exported, select the PCB board; click the paper icon named 'File'; and click 'Export...'. For technical details, the file is set to '.fbrd'.

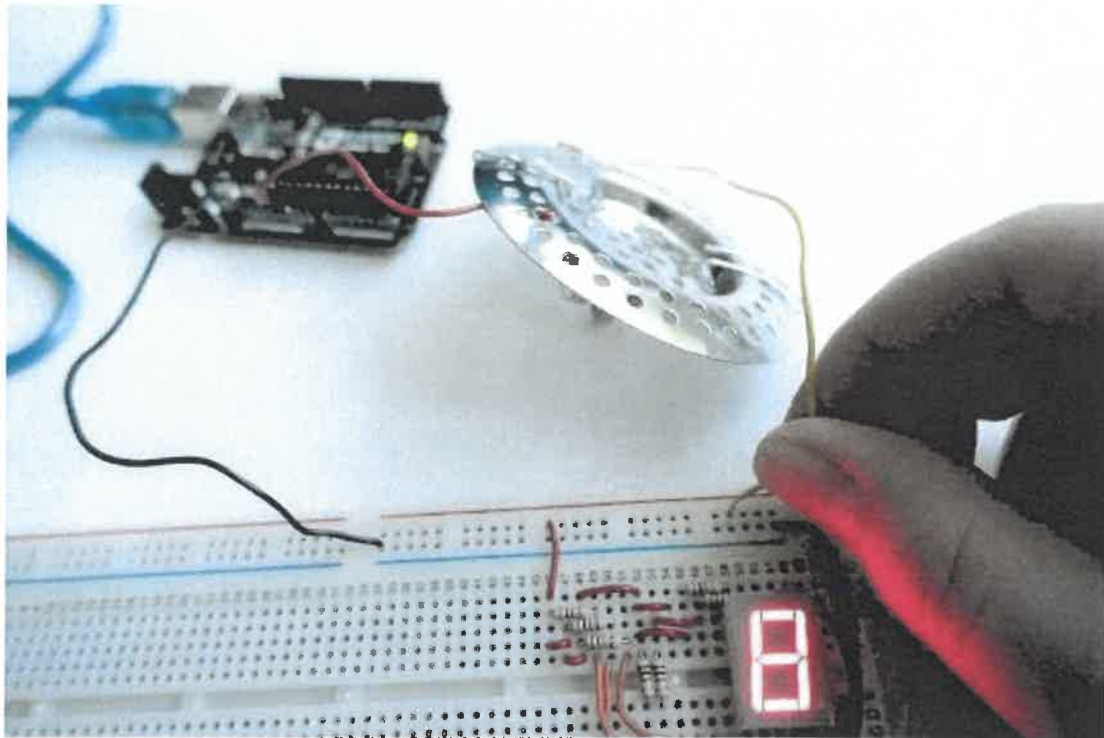
Proof of 3D-Printed Model



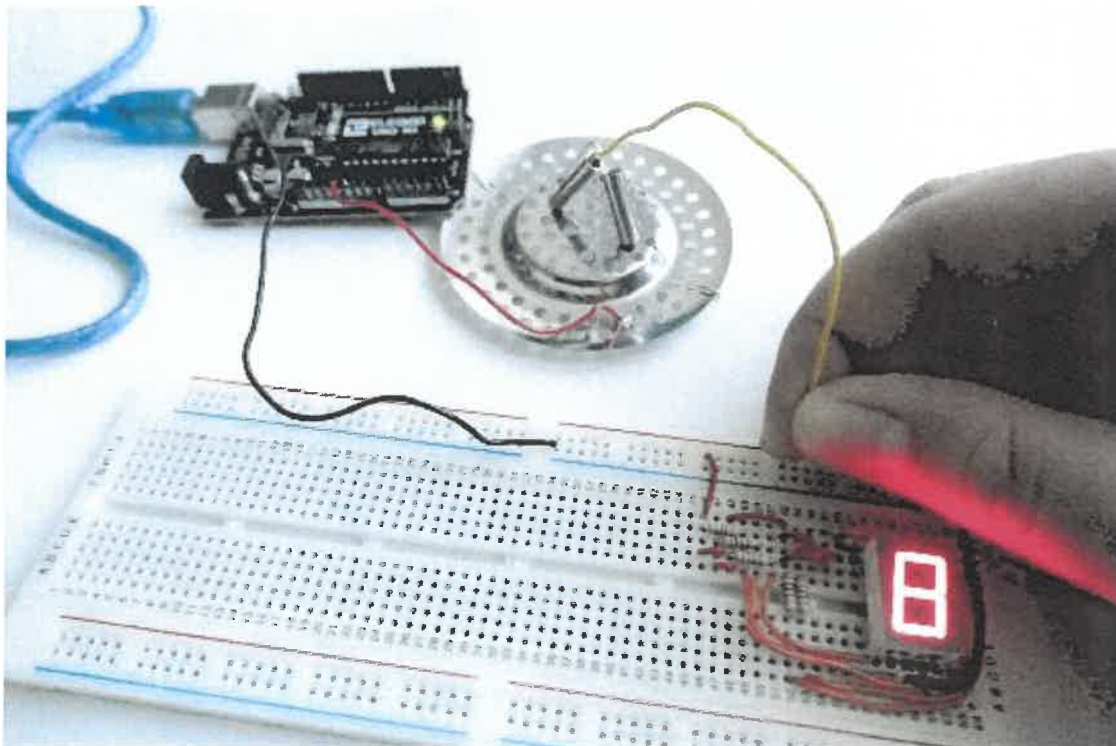
As one of the smaller products was successful in being 3D-printed, two images of the wire holder are provided as proof of development.

Proof of Wire Top Holder (makeshift, not Fusion 360)

Due to having acquired a sink strainer (which emulates the developing country creativity of creating makeshift products), testing of electrical conduction is to be carried out for proof of overall experimental success.



The result of connecting a sink strainer with electricity is proven successful (above image), as is connecting it with the metallic contacts developed (below image).



To produce metallic contacts, see page 56.

Production of Wire Top Holder in Images

Repeat with every two subsequent holes 2nd closest group to central hole.



Makeshift Demonstration of Wire Game



Not visualised above, the holding stand is a 100 mm/4" paint roller and is held in place by a heavy object. Roughly attached to it is the top wire holder, and it is relatively wide for a wire game obstacle.

To make sure the wire game is easy-to-medium as an interactive game, the hook's circumference should be no less than a saucer (a plate small enough to hold a teacup).

Further assembly and implementation are to be carried out after 12 days.

Development Reflections

The following ideas plus those in the images relevant to the document's chapter have been voided during development:

- Timer (three more 7-segment displays and 74HC93 + CD4511 combos)
- Metal components being shaped and designed through AutoDesk Fusion 360
 - Wire Top Holder Claw
 - Plastic Cover for Wire Top Holding Rod
- Implementation of DIP-switch to select options before gameplay
- Instruction Leaflet
- Twist Lock for Wire Top Holder
- Wire in the Wire Game being set to a specific shape
i.e: "Hamiltonian-pattern cube of 4 cubic meters or a Hamiltonian tesseract"
- Implementation of binding screws and a comparator circuit.

Some design hypotheses that ended up working include:

- The decision to source power through the hook and let all in-game metal surfaces run through pull-down resistors (albeit not knowing the term during research).
- Different noises emitted through the beeper via. Arduino UNO's C++ code.
- The motor rotating the custom wire shape.

Voided Ideas

Timer

The timer aspect of the wire game led towards a perceived complication when consideration towards development was done, with the conflict of external matters preventing it from being fully realised.

AutoDesk Fusion 360 Components – Wire Top Holder Claw + Rod Plastic Cover

Since illustrations for ideas have been developed within (see Initial Report, pages 4 and 5), later development came to the conclusion that the wire top holder claw cannot be shaped through AutoDesk Fusion 360 due to time constraints.

The plastic cover for rod is also voided as it misinterprets what the software can perform. Making a plastic cover for the rod often requires a more industrial method of modification. Watch how cars' doors are painted for comparison.

DIP Switch Settings

Having to implement the DIP switches rolled over to the point where a primitive rendition of telecommunication has to be programmed and constructed. And it could have led to a waste of digital I/O pins of the Arduino used for the project i.e: bit combinations.

Instruction Leaflet

The instruction leaflet can take a long time to finalise entire designs to add illustrations for while building with the project. It can lead to almost twice the amount of time needed to finish the assignment.

Twist/Flip Lock for Wire Top Holder

Since not enough research was brought towards locking the wire top holder from moving during gameplay, an alternative for serving such purpose can involve sticky tack instead of more components to 3D print through AutoDesk Fusion 360.

Specific Shape(s) for Wire Game

Initially, the design was going to include a complex design made from bending a steel rod. But due to the same time constraints, makeshift wire with a drilled steel ball could do as a substitution for a wire whilst flexibility of its shape can be easily offered.

Binding Screws and a Comparator Circuit

The binding screw was in consideration towards the printing of the Wire Top Holder Claw. But it is then voided due to the same time constraint reason.

The comparator circuit was towards initial concern of noisy signals when the hook touches either wire or win contact. A pull-down resistor circuit for each is immediately substituting once realised. The substitution is then seen as a successful hypothesis.

Final Design Details

There was mentioned that laser and magnetic interference tests were to be conducted once the product is finished. But considering the time constraints obstructing past full completion, such tests may not be able to be carried out.

Once the plastic casing designed in Fusion 360 was developed, there was no consideration for how power is to be supplied physically through USB connection. The product lacks a hole in which a USB socket can fit through the wire game casing. To fix this, drilling a wide-enough hole in the appropriate spot allows 5-volt connection for the wire game to start.

The wire game originally was imagined to be more than half the final height (originally ~7cm tall, but now ~10cm) and included some of the ideas mentioned that were then voided for brief reasons (see Development Reflections - Voided Ideas).

Within two days before deadline, further components including: spinning wire, more parts of the wire game casing, PCB for wire game, etc. are to not be assembled due to time constraints.

Since the project is described to be of educational purposes, hand-motor skills can be developed for the participating demographic as well as their understanding of current flow with contacting materials involved. The project can also be able to make the demographic more interested towards electrical principles and other technological concepts through such experience.

Evaluation of Final Product Quality Assurance

Considering the demographic is of smaller, open-minded students of a young-enough age, there will be numerous quality assurances to consider.

By using bevelling tools in Fusion 360, the product will contain fewer sharp edges to prevent sudden cuts being inflicted that can cause first-aid related events to occur.

Reminders for not wearing metallic clothing or accessories is recommended due to relevance of metallic conduct that can cause sudden injuries towards such participants.

To acquire rubber feet for the project ensures that the project does not slip and break from high platforms.

While developing the 3D model on Fusion 360, consistency to implement jigsaw fitting per component piece should be carried out instead of starting with the unrefined method and performing the complete procedure for the rest. This should ensure that confidence towards fitting component pieces is self-verified to be properly fit before batch production.

Recommendations

To improve against time constraints, the opportunity to purchase a bigger 3D printer and access to support to help print out the entire project casing makes development much easier.

Conclusions

Development started with regular progress being carried out. Mid-year development ended up being slow, but the final three months took on greater priority towards the assignment with barely justified obstruction near the end of development.

The breadboard prototype of the intended project did succeed functionally after many iterations.

However, implementing the successful code into the 3D printed arrangements was enforced to be set to termination due to time constraints. Only one component was successfully 3D printed, and the rest is left abandoned for the same reason.

References

ALkhateeb, A., 2020. *Controlling and Positioning Stepper Motor using LED's*. [Online]
Available at: <https://www.tinkercad.com/things/i4M2ImckiL3-controlling-and-positioning-stepper-motor-using-leds>

[Accessed 31 March 2023].

Khiatani, J. V., 2022. *Toy Regulations and Standards in the UK: An Essential Guide*. [Online]

Available at: <https://www.compliancegate.com/toy-regulations-united-kingdom/>

[Accessed 9 December 2022].

Maxfield, C., 2022. *Arduinos and Solderless Breadboards*. [Online]

Available at: <https://www.clivemaxfield.com/coolbeans/arduinos-and-solderless-breadboards/>

[Accessed 17 May 2023].

Programming Electronics Academy, 2021. *WHAT IS SERIAL.BEGIN(9600)?*. [Online]

Available at: <https://www.programmingelectronics.com/serial-begin-9600/>

[Accessed 17 March 2023].

Simple Projects, 2018. *Arduino Unipolar Stepper Motor Control*. [Online]

Available at: <https://simple-circuit.com/arduino-unipolar-stepper-motor-control/>

[Accessed 31 March 2023].

Append – Project Report Appendices

Contents

1. Instructions for Functions	3
1.1 Fusion 360	3
1.1.1 Downloading Fusion 360.....	3
1.1.2 How to View All Stock Libraries in Fusion 360.....	4
1.1.3 How to Produce a Rod Shape	5
1.1.4 How to Render a Design in Fusion 360.....	9
1.1.5 How to Change Physical Material of a Design.....	10
1.1.6 How to Create a Wire Stand that should Rotate with the Step Motor	15
1.1.7 How to Push a Circuit Design into 3D PCB.....	22
1.1.8 How to Place Text on Circuit Design	23
1.1.9 How to Insert Components to a New Project	24
1.2 Arduino IDE	25
1.2.1 How to Demonstrate a Basic ‘Blink’ program on Arduino UNO using Arduino IDE	25
1.2.2 How to transfer TinkerCAD code to Arduino UNO.....	28
2. Components in Use of Product	29
2.1 Initials in Integrated Circuits.....	29
Texas Instruments - Important Notice and Disclaimer	30
2.2 Integrated Circuits (ICs).....	31
2.2.1 4511 ICs – 4-Bit BCD-to-7 Segment Latch Decoder Drivers.....	31
2.2.2 7400 ICs – Quad 2-input NAND Gate	32
2.2.3 7493 ICs – 4-bit Binary Counter	33
2.3 Other Components.....	33
2.3.1. 7-Segment Display	33
3. Tables involved with Project.....	34
3.1 Prototype’s Connections per Component.....	34
3.1.1 CD4511 – BCD 4-Bit Decoder.....	34
3.1.2 7493 – 4-Bit Binary Counter	34
3.2 Pin Tables in Varying Formats	35
3.2.1 7400 and 7408 Connections – NAND & AND Gates	35
3.2.2 Time Increment/Decrement and Contact/Mistake Increment.....	35
3.3 DIP Settings [Voided/Cancelled].....	36
3.4 Pin Correspondence for Project	37

4. Non-Final Tests/Arrangements during Development	38
4.1 TinkerCAD - 4-Bit Segment Base-10 Display Test	38
4.2 TinkerCAD - Adding Three More 7-segment Displays	43
4.3 Fusion 360 - Marble and Metallic Contact Visualisation.....	45
5. Code Batches – Full Code Variants used in Arduino UNO	46
5.1	46
5.2.....	48
5.3.....	50
References.....	52
Image References	52
Video References.....	53
Forum References	53

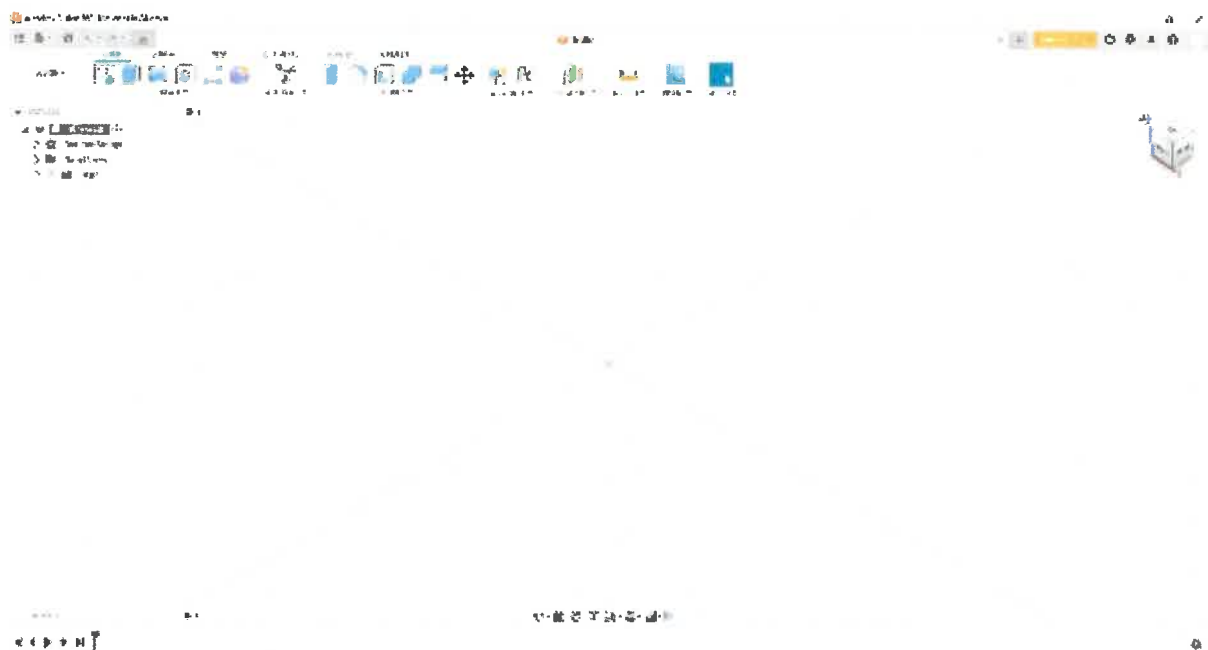
1. Instructions for Functions

1.1 Fusion 360

1.1.1 Downloading Fusion 360

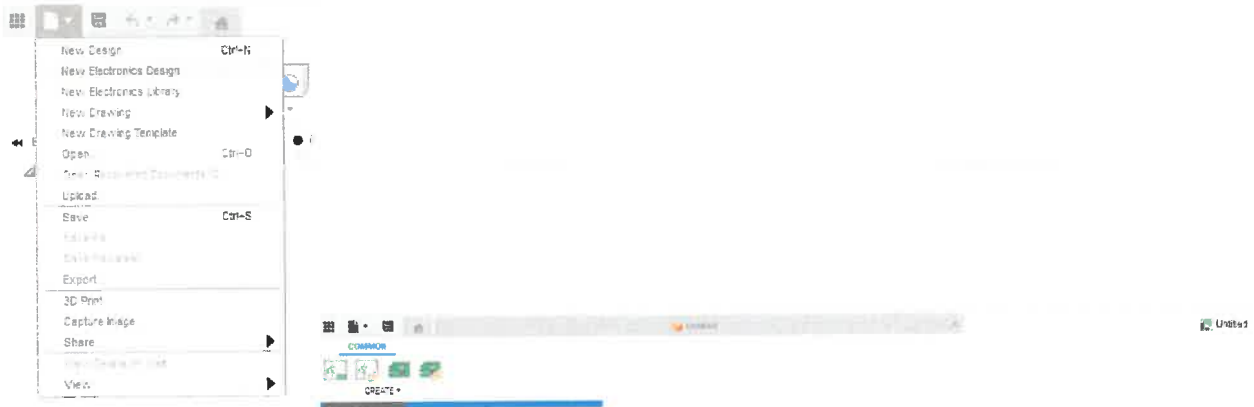
In the Autodesk website and selecting Fusion 360, a free trial can be downloadable for personal development if being signed-up is carried out. But in due location of education's demand, providing documentation (after clicking the 'Get Started' button by entering 'Fusion 360 student license' on the website's search bar) such as: school transcript, school-issued confirmation letter, school ID, etc. are required for extended duration of use.

Once the Autodesk installation wizard is installed to the computer, select it and follow its steps so to download the Fusion 360 software.

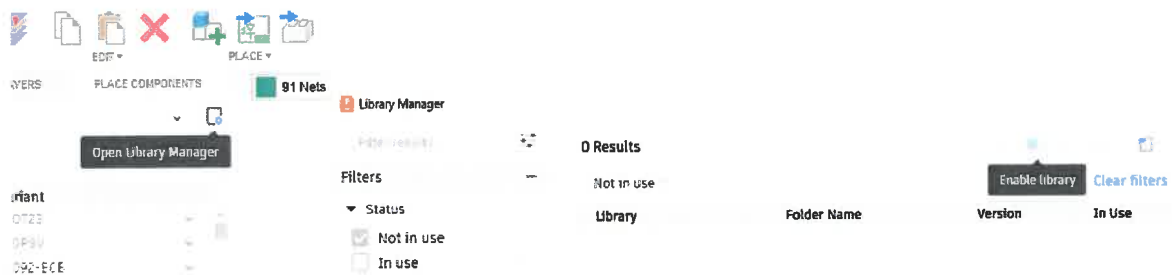


After signing up through the software, a pop-up should display ways of going to certain areas of the software after entering for the first time. The rest of this section is not logged as results may vary.

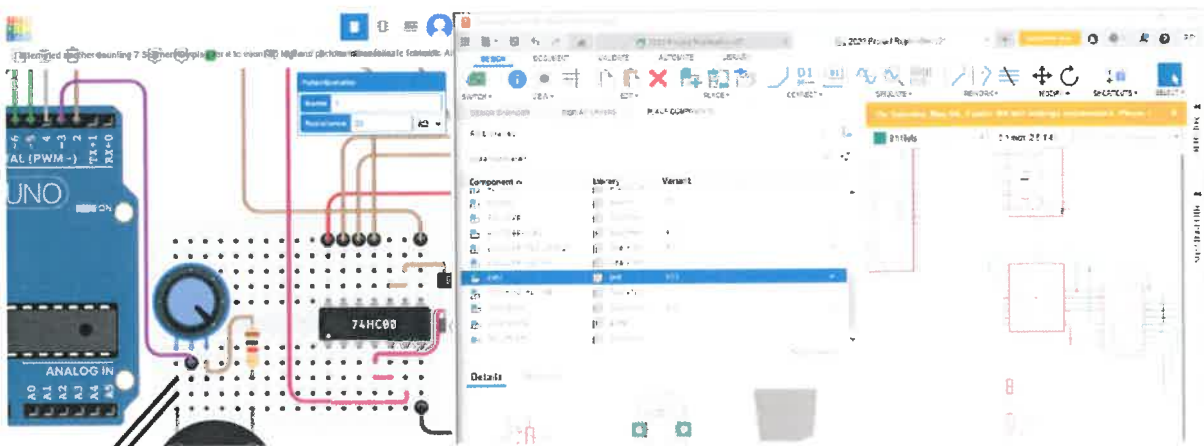
1.1.2 How to View All Stock Libraries in Fusion 360



Click on the paper icon's drop-down arrow, click 'New Electronics Design' and then click on the first green icon in the set of four choices from the 'Create' tab.



To access all components from the Schematic screen, select the 'Open Library Manager' button. Enable the 'Not in use' status and select the second option of four icons above the libraries named 'Enable library'.

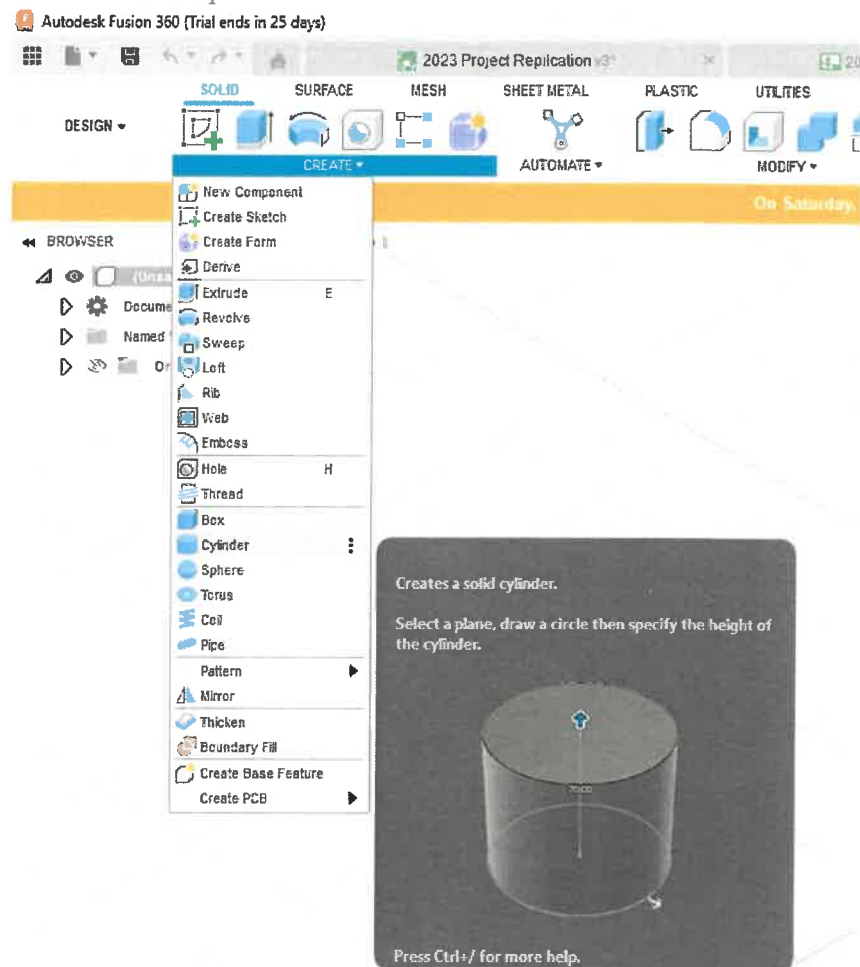


You can select components by finding them in the 'Place Components' tab. For convenience, search the component by the 'Filter...' bar or search the component's library on the search bar above the 'Filter...' bar.

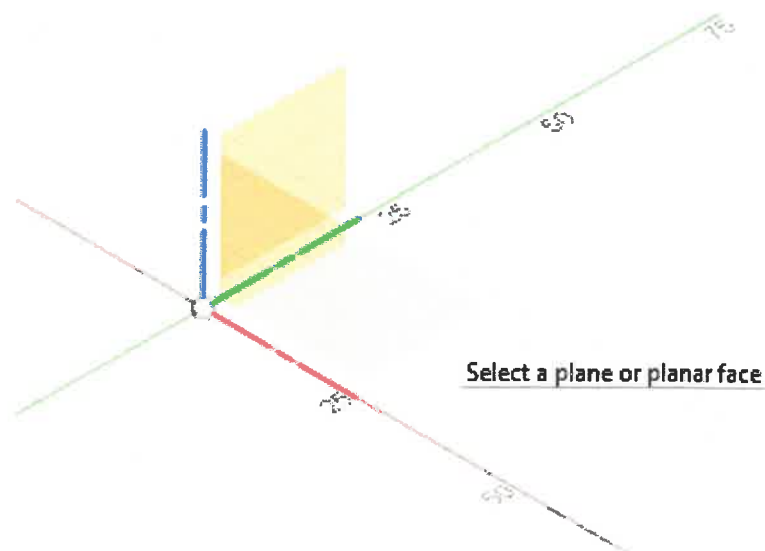
Select a variant if a drop-down arrow is present for it and drag the component to the right-hand side of the software's screen to place the component.

In the above example, a potentiometer is to be found in the list of components since the document's project is due to acquiring one.

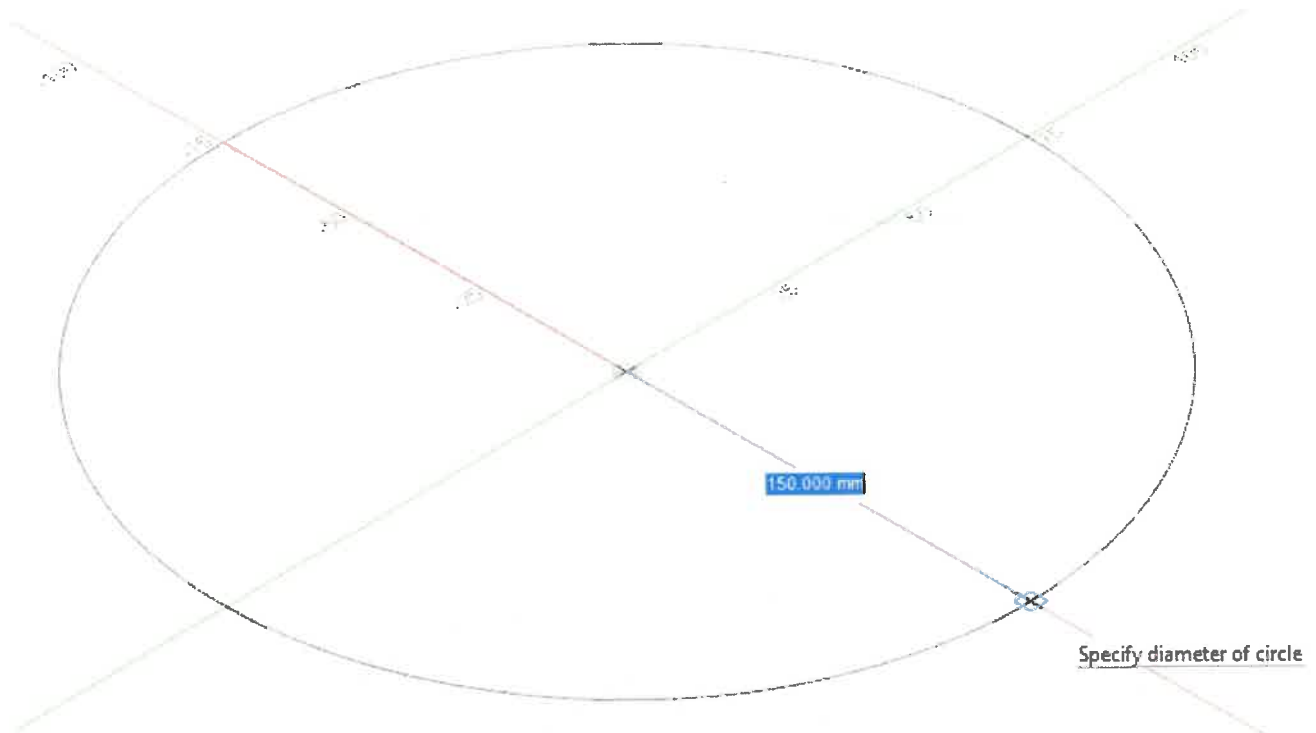
1.1.3 How to Produce a Rod Shape



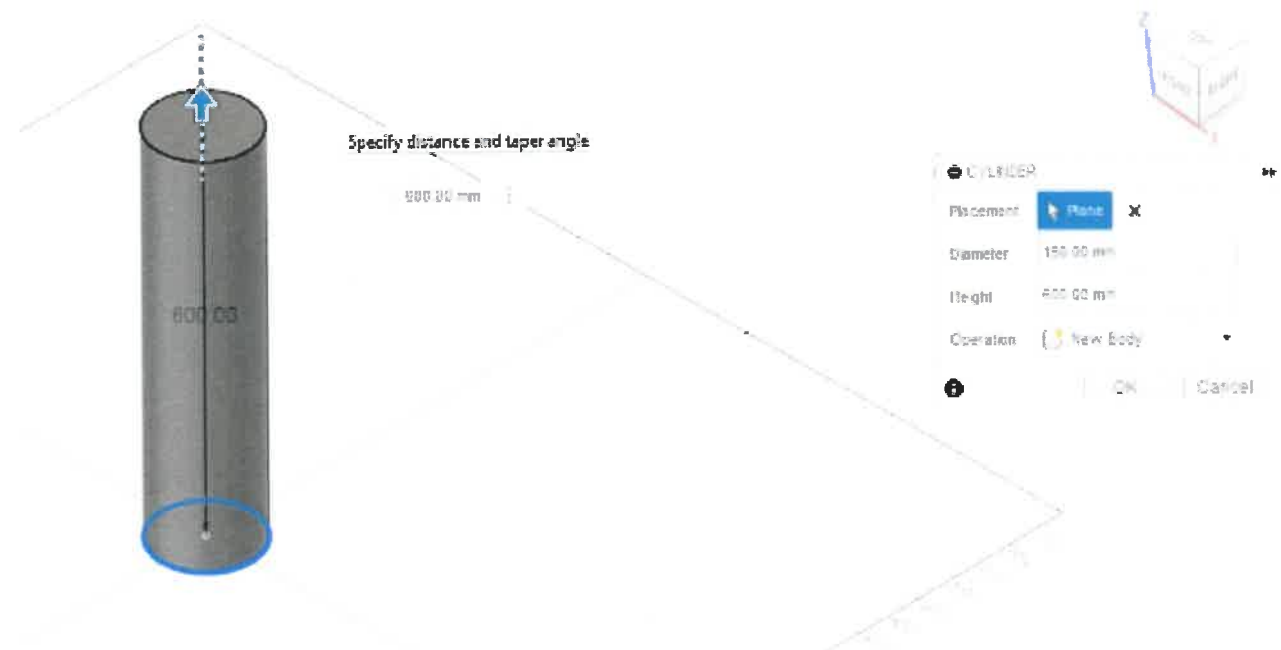
In the 'Design' screen, click on the 'Create' drop-down button then select 'Cylinder'.



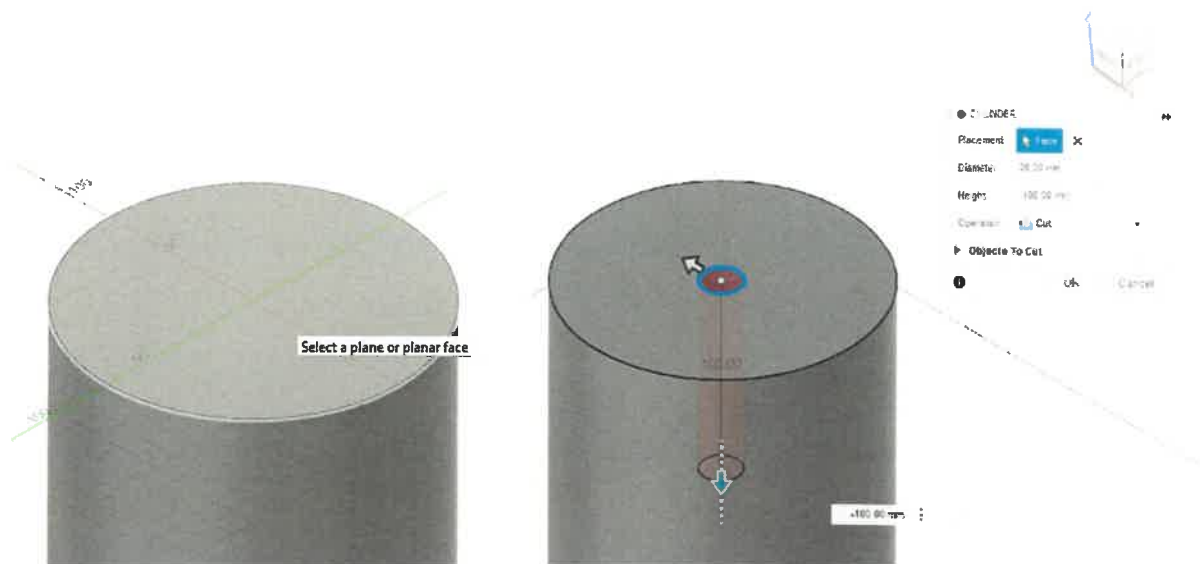
Once 'Cylinder' is selected, select an appropriate plane or planar face.



Click a central point for the circle to generate on the selected plane or planar face, then specify the diameter of the circle by clicking how far from the central point the circumference is.



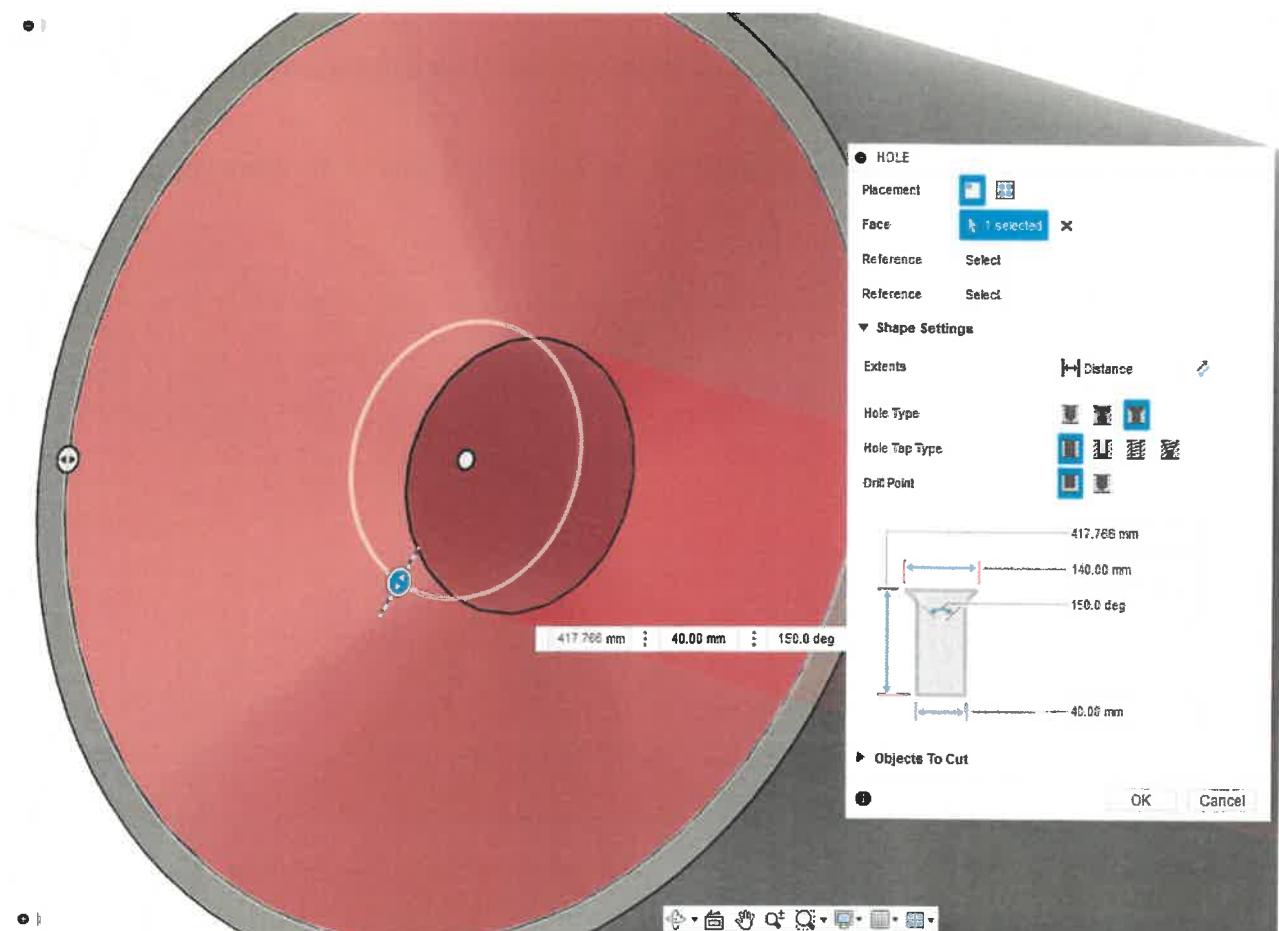
Once a circle is generated, drag the up arrow to a suitable height. For the tutorial's instance, set to 1500.00 mm. Click OK from the 'Cylinder' window once done.



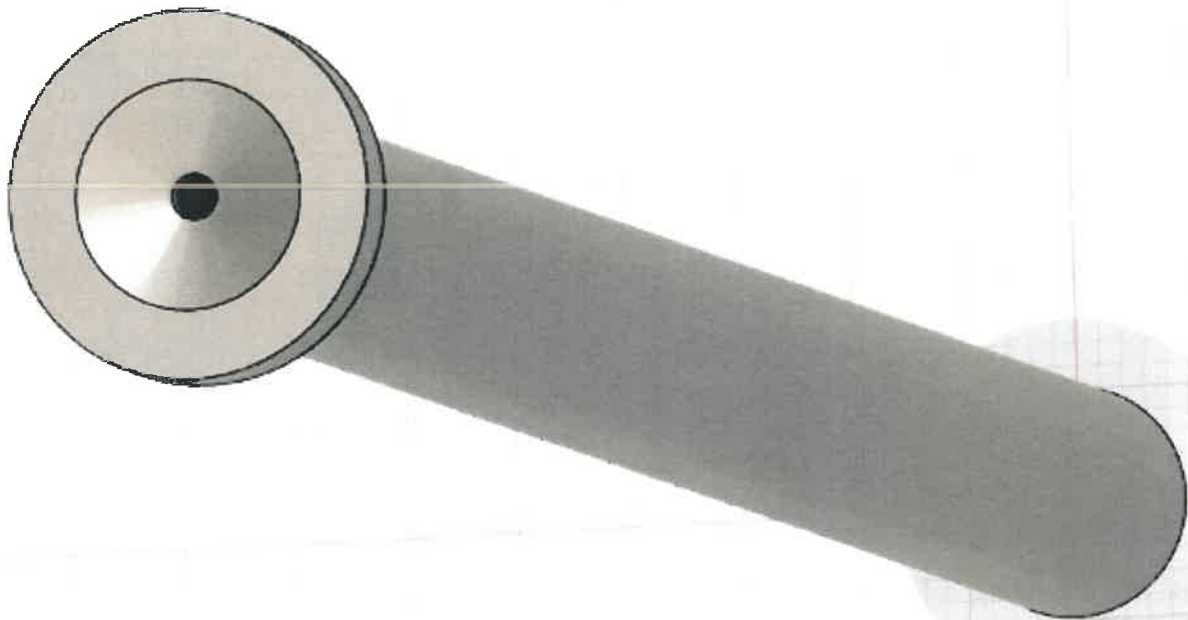
To create a protruding hole through the cylinder, click on the cylinder's face then select the 'Cylinder' option again.

Select its central point and size diameter to 20.00 mm.

Drag the vertical arrow so that it extends up to the ends of the base cylinder.



Alternatively, a hole can be created by clicking the 'Hole' function above the 'Create' tab. Perform the above instructions, then you also have the ability to create a hollow hole tap.



An example rod produced before the tutorial was designed as the above image. An extra cylinder body will be needed to produce the above rod.

Perform another creation of a cylinder body central to circular face and expand to a slightly larger diameter.

Make another cylinder for cutting the second body so that it can maintain the details of the first cylinder body.

1.1.4 How to Render a Design in Fusion 360

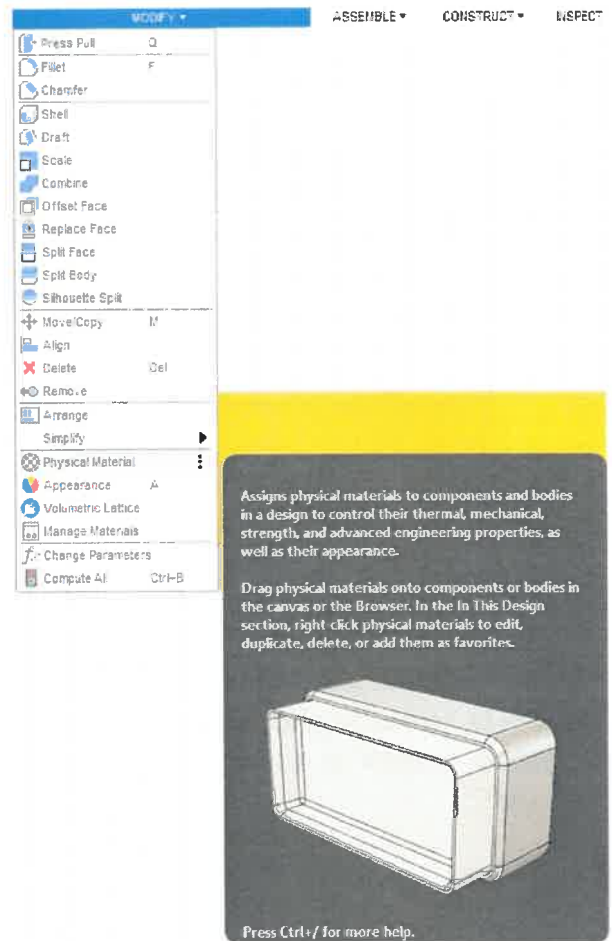
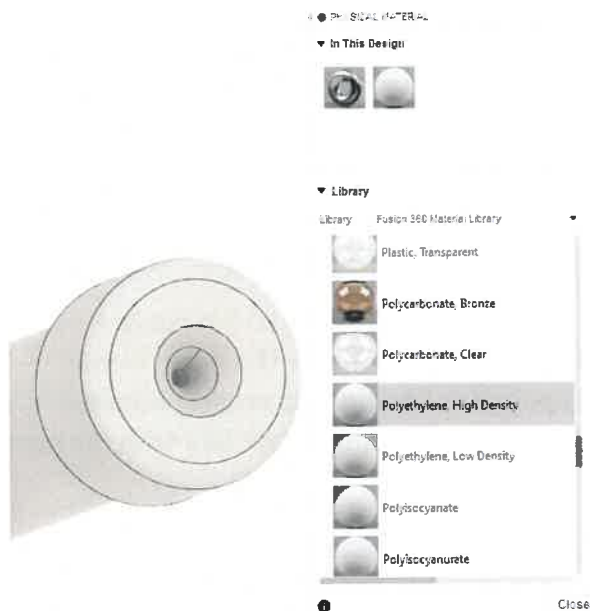


To 'render' the resulting product, click on the large button at the top-left and select 'Render'.

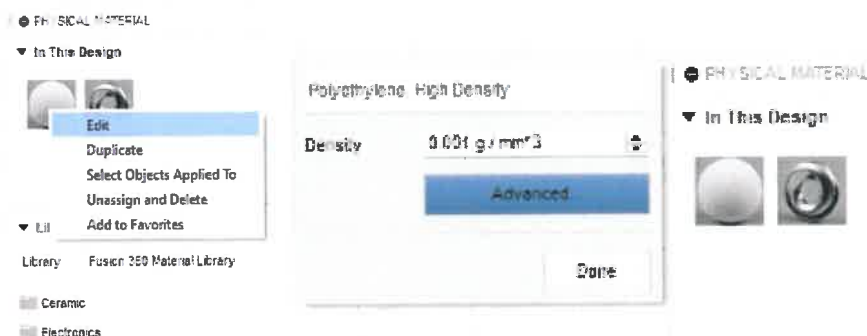
For the design to have different material faces, open the library by clicking on the rainbow wheel. Select a material from the library window. In the example above, 'Polymer (Kapton)' is used. However, said example material is used for industrial purposes like flexible printed circuitry. Consider 'Render' as a visual demonstration as to what the product might look like with the materials selected.

1.1.5 How to Change Physical Material of a Design
To select physical material for the rod, click on the 'Modify' drop-down tab then click on 'Physical Material'.

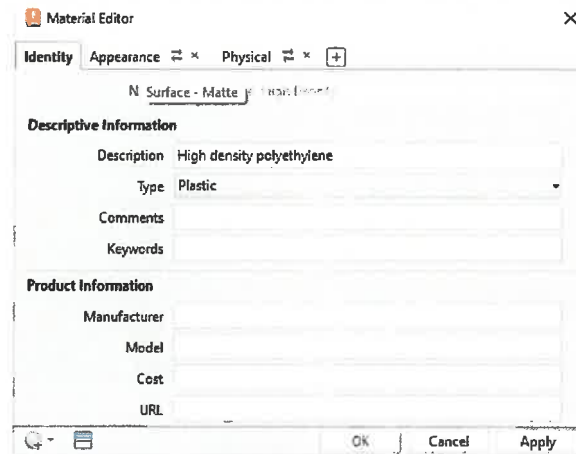
Drag and drop a material that should fit the use of the rod regarding the overall purpose of the entire product.



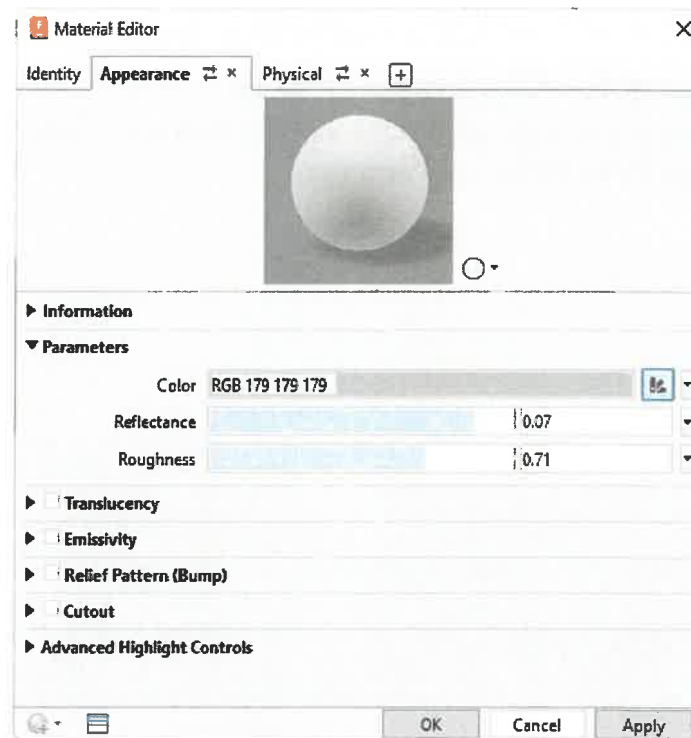
To colour the selected material, right-click on the material and click 'Edit'.



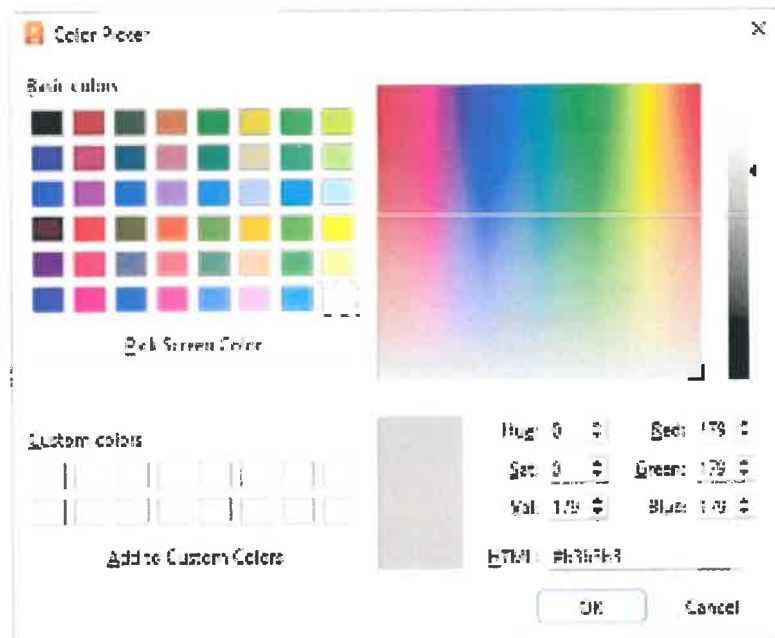
Material name and density are viewable in the provided window. Click on 'Advanced...' for more.



Select 'Appearance' from the row of tabs.

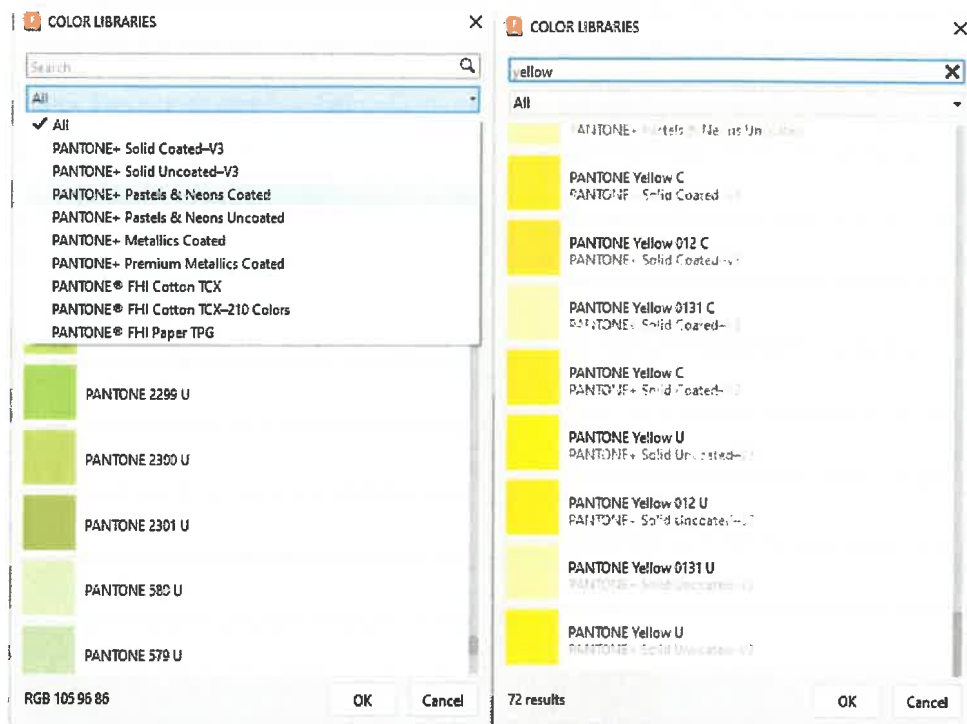


Either click on the bar that shows the 'Color' to open a 'Color Picker'. Or click on the icon to the bar's right to open the 'Pantone Color Libraries'.



If selected 'Color Picker', freely choose a colour of choice.

In the document's example, a specific yellow (PANTONE Yellow C, PANTONE+ Solid Coated-V3) is selected from 'Pantone Color Picker'; see next page.

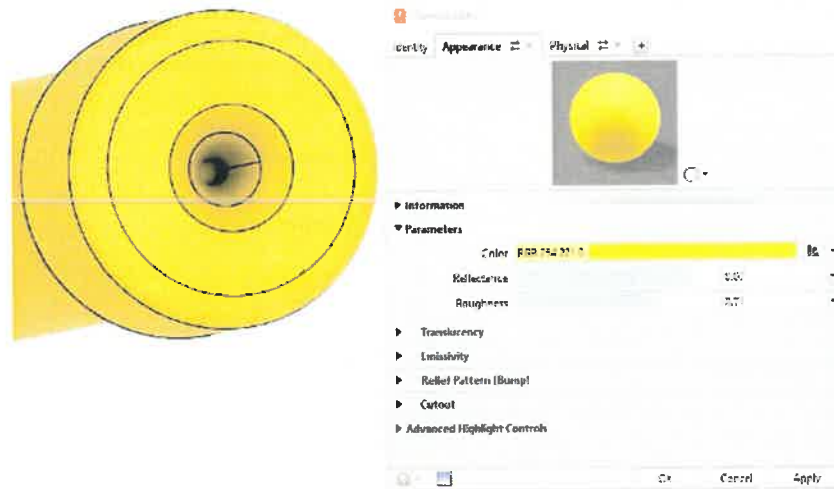


In 'Pantone Color Picker', the window is named 'COLOR LIBRARIES'.

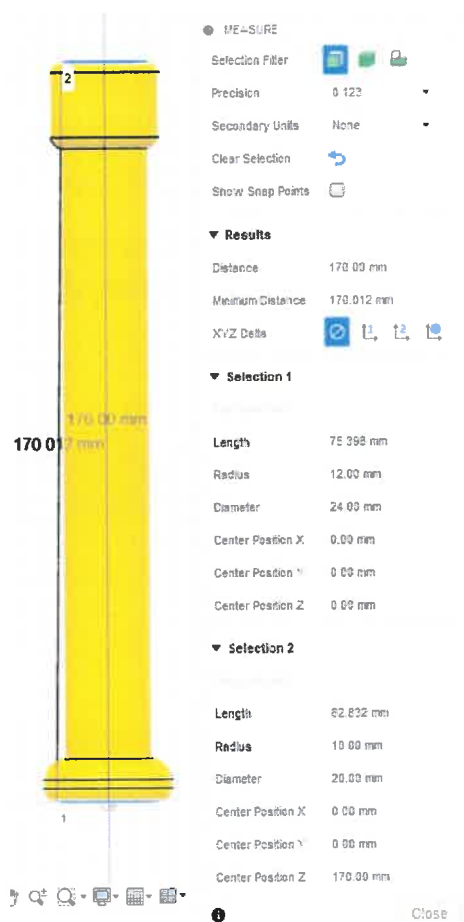
The option to specify a category of colours is available, as well as a search bar to find a specific colour. Use both to specify color and appropriate estimation of paint type.



See next page for result of colour change.

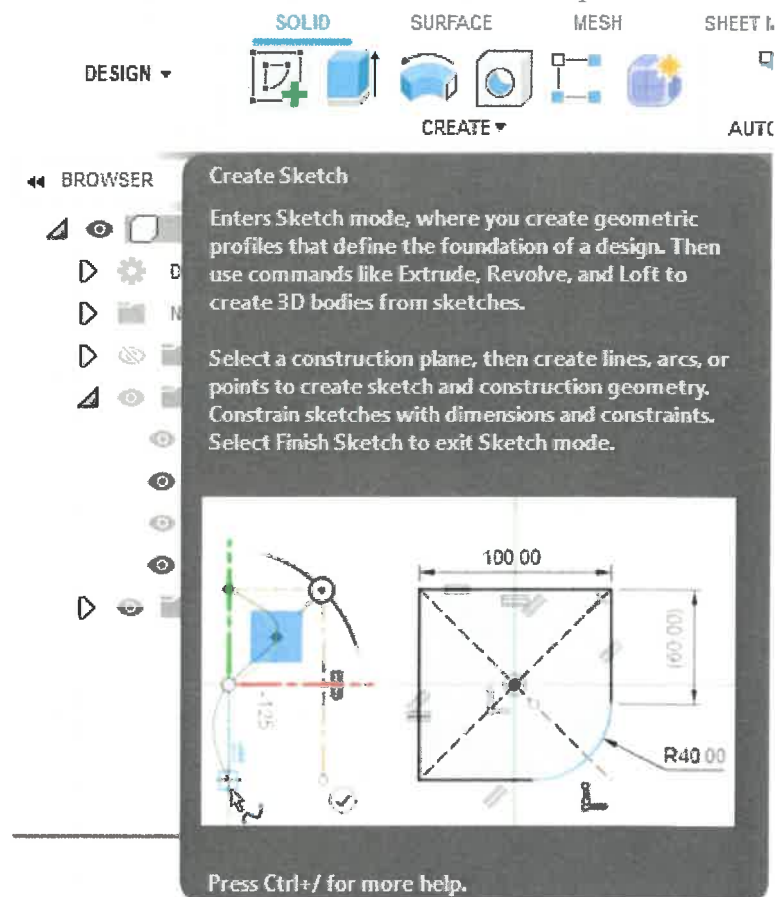


The resulting change should change and paint the rod into a yellow plastic.



Here is what the rod would look from the steps given.

1.1.6 How to Create a Wire Stand that should Rotate with the Step Motor

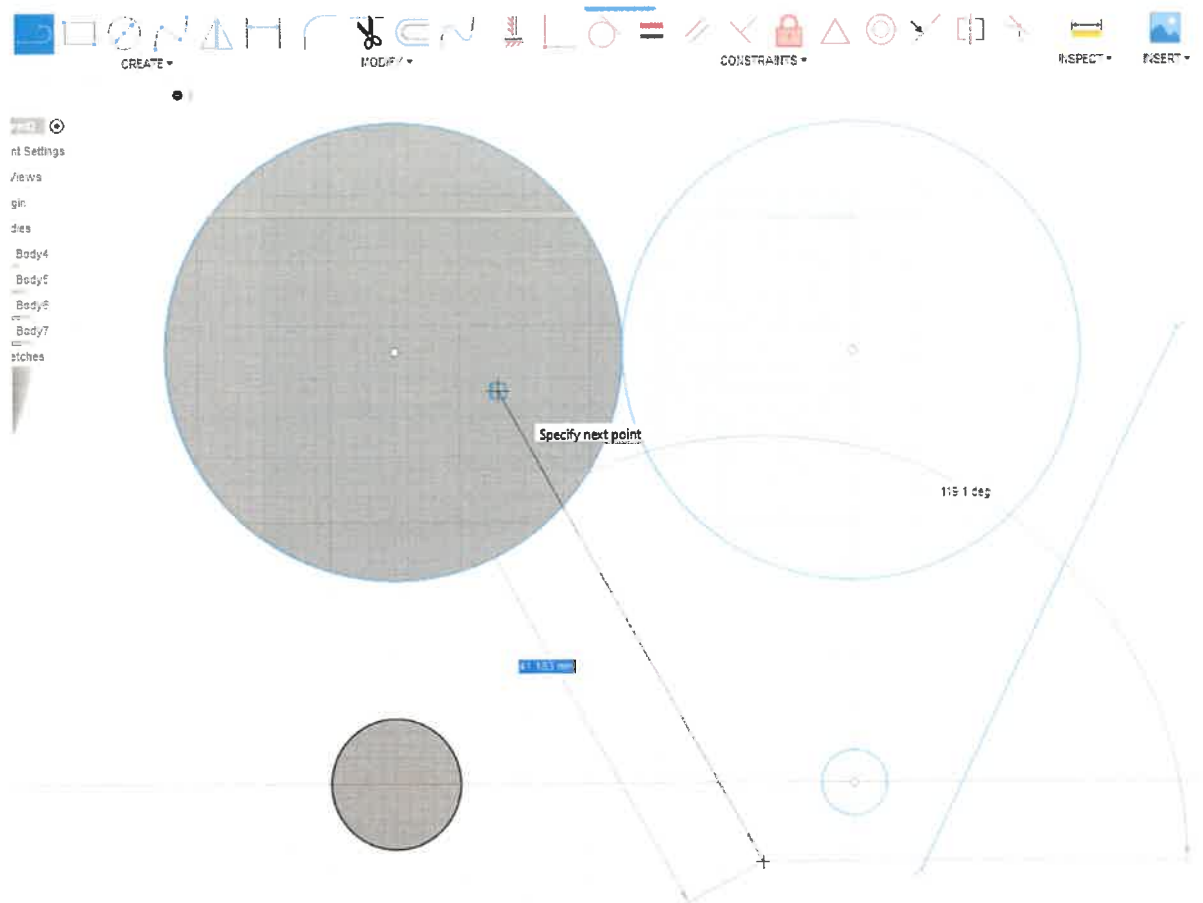


To sketch a new shape in Fusion 360, select the first icon above the 'Create' drop-down bar.

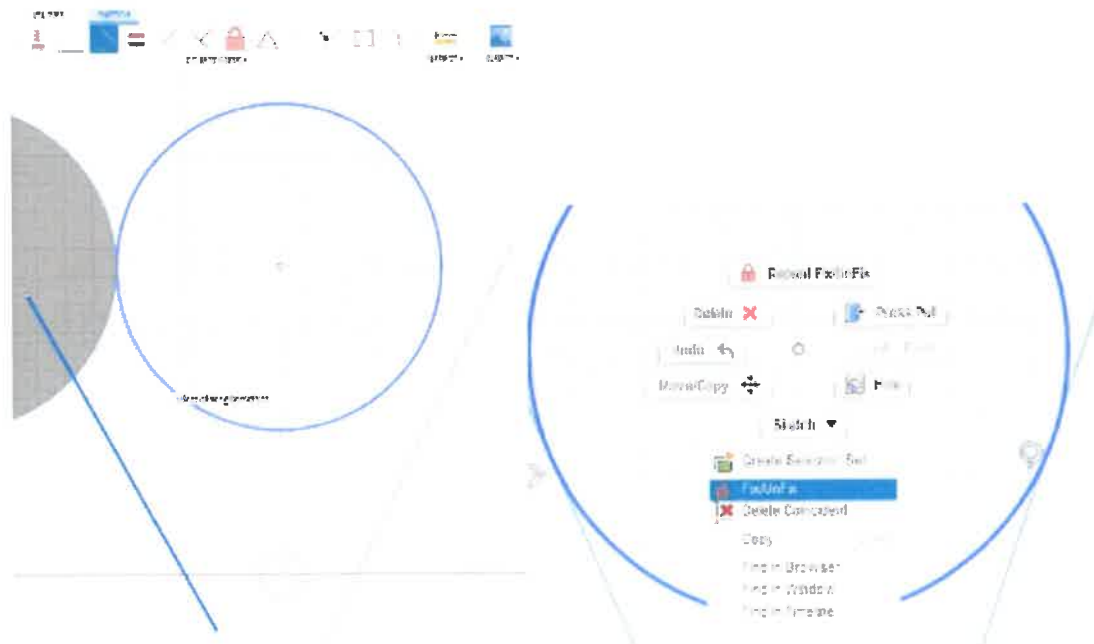


The stand should be of a simple pear shape, so select the 'Center Diameter Circle' icon.

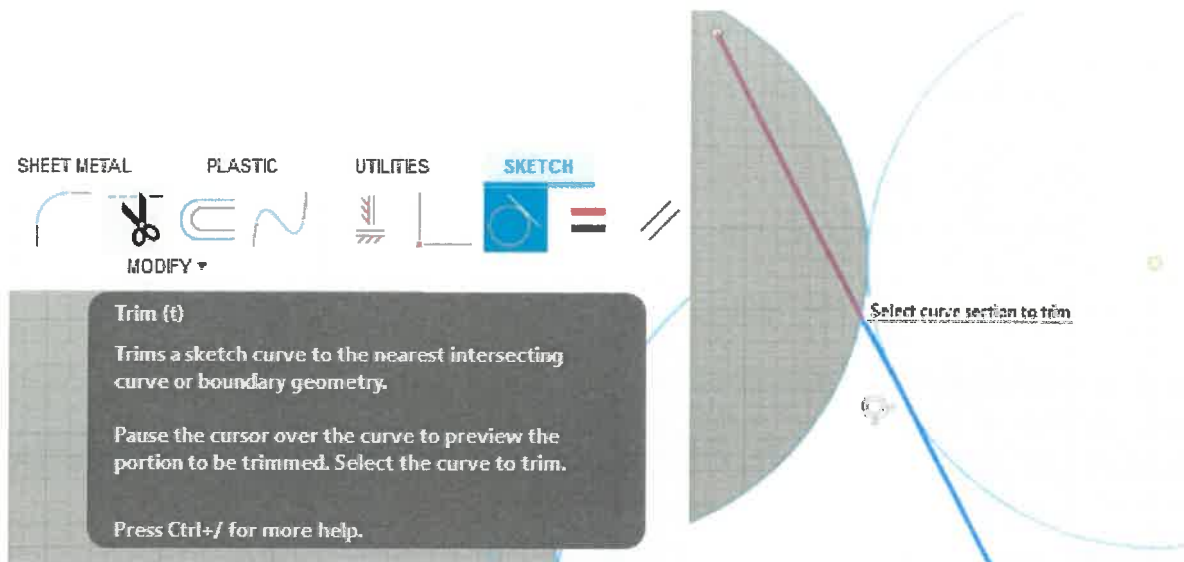
Create two different-sized circles near, but not touching, each other. In the document's instance, a 5mm circle and 35mm circle with centers 33mm apart.



Draw two lines that should be longer than the perceived distance that both circles' tangents meet by clicking on the first icon above the 'Create' drop-down bar.

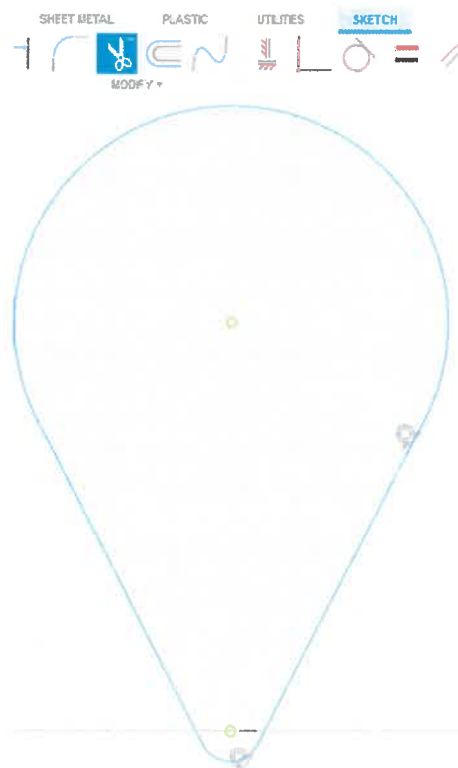


Right-click on the centers of the circles and select 'Fix/Unfix' to ensure that the circles do not depart from their original parameters, then select the third icon above 'Constraints'. Connect respective lines to both circles on each side.



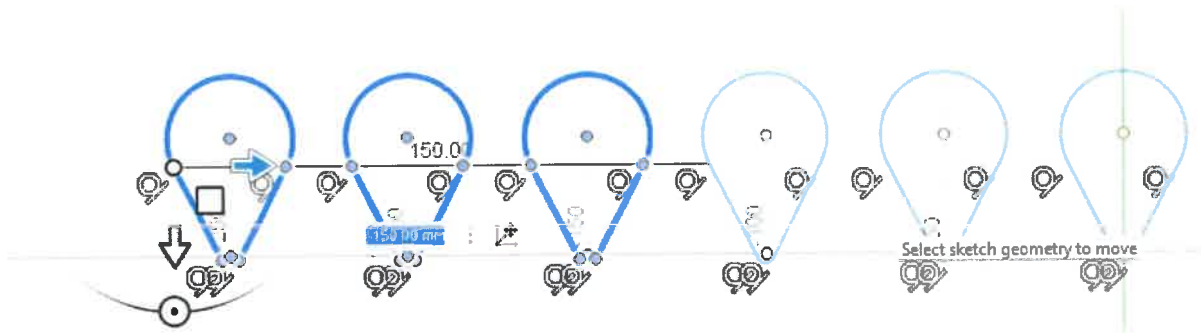
To trim out unwanted lengths, select the 'Trim' icon above the 'Modify' drop-down.

Select sections that fall as an obstacle against the 3D-transforming process that happens later on. Sections highlighted in purple are those that are to be trimmed.



The wire stand should look like the lowermost image. Unfix via. 'Fix/Unfix' to allow duplication. And as a warning, do not shift or relocate the sketch once exiting 'Sketch Mode' in case errors arise. For reasons to produce variations, remain in 'Create Sketch' and follow all remaining steps in the tutorial.

And note that overlapping lines can be barely visible through the thickness between sketch lines.



Select your entire sketch, leaving none of its lines or area behind.

Either right-click and select 'Copy' in the right-click list, or press 'Ctrl' and 'C'.

Then press 'Ctrl' and 'V' on the keyboard to produce a duplicate of the wire stand and drag it in a convenient area. Repeat duplication for desired quantity.



To create a notch for the wire stand, there are two ways that sketching the stand can be done.

1. Select the second icon above 'Create' while in sketch mode.

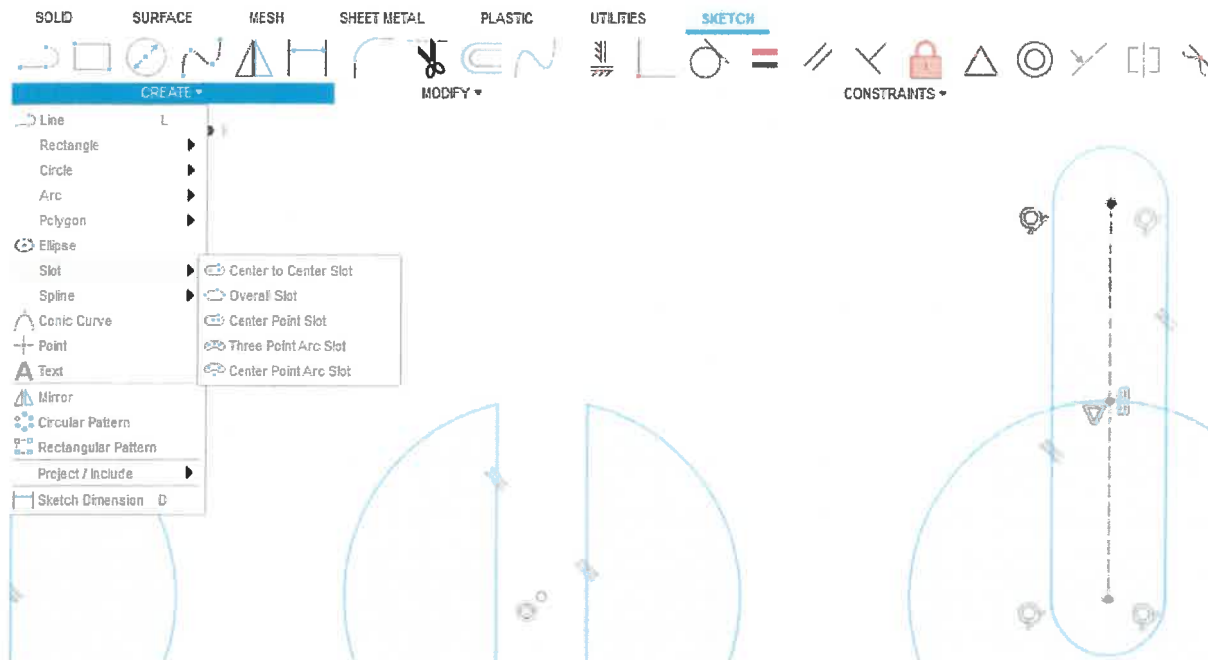
Locate the box so that its top side touches the highest point of the wire stand and align vertically-central.

Skew the left and right sides of the box depending on the thickness of the wire while horizontally-aligned.

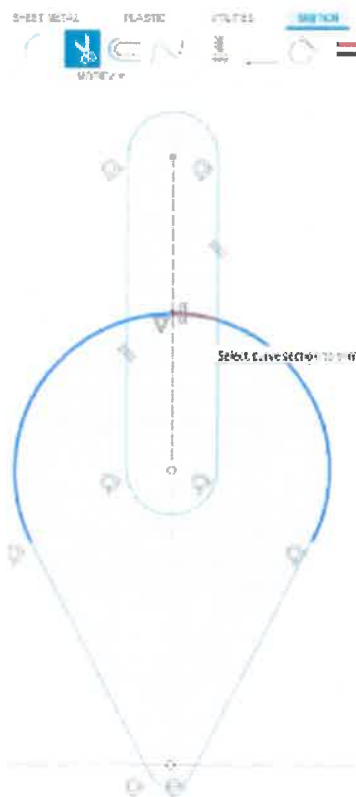
Stretch the bottom side of the box to determine how deep should the wire stick in the stand. Make sure to give some space for the stand in case an area of it is too thin.

Select the 'Center Diameter Circle' and click in the middle of where the selected core removal point is produced. Size the circle from the point to the entire width of the box cut.

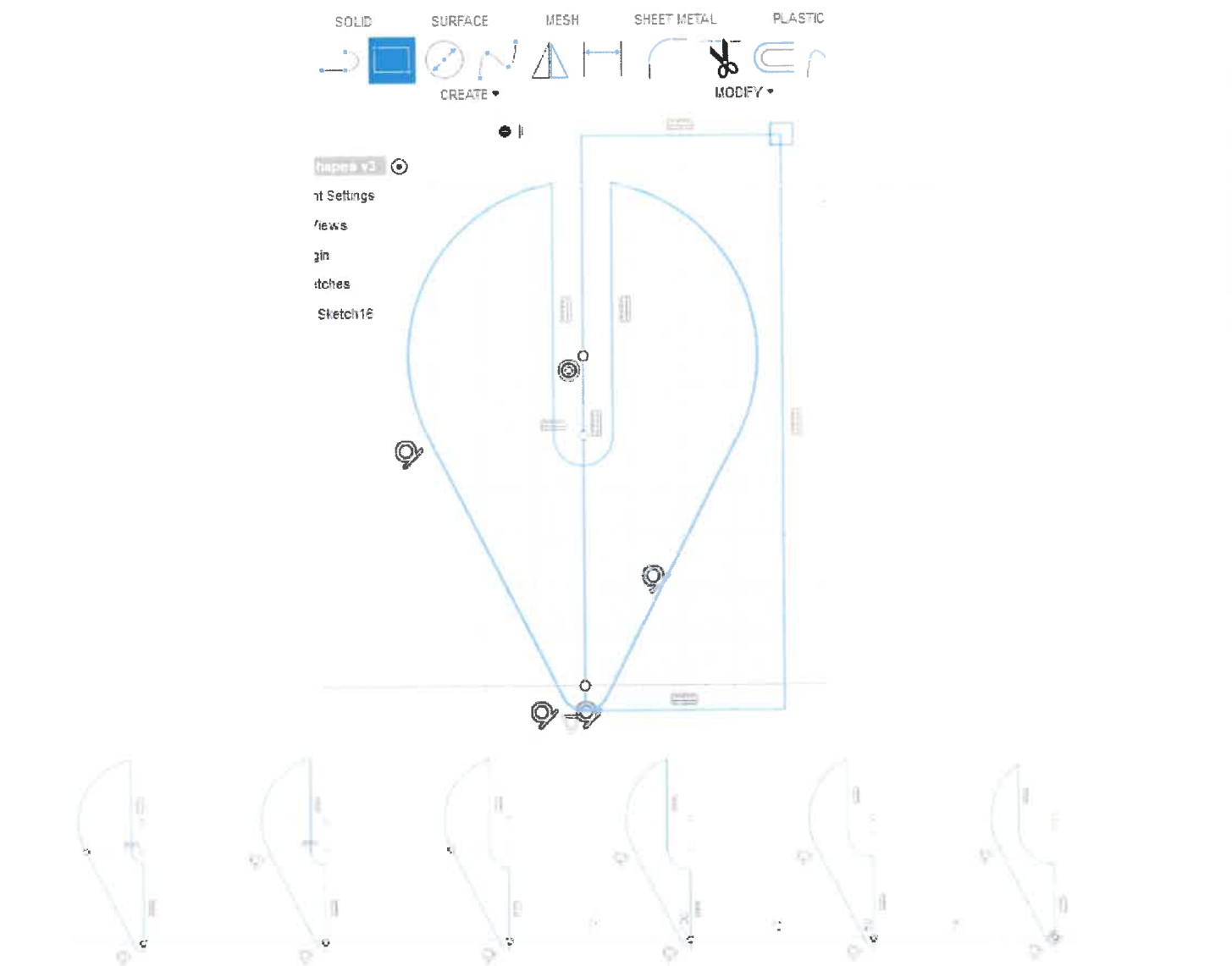
Select the 'Trim' option to remove unnecessary lines from both the box and stand.



2. Select the 'Create' drop-down bar and hover over 'Slot'. Select either 'Center to Center Slot' or 'Center Point Slot'. Perform similar peak-to-core selection like in the first method.

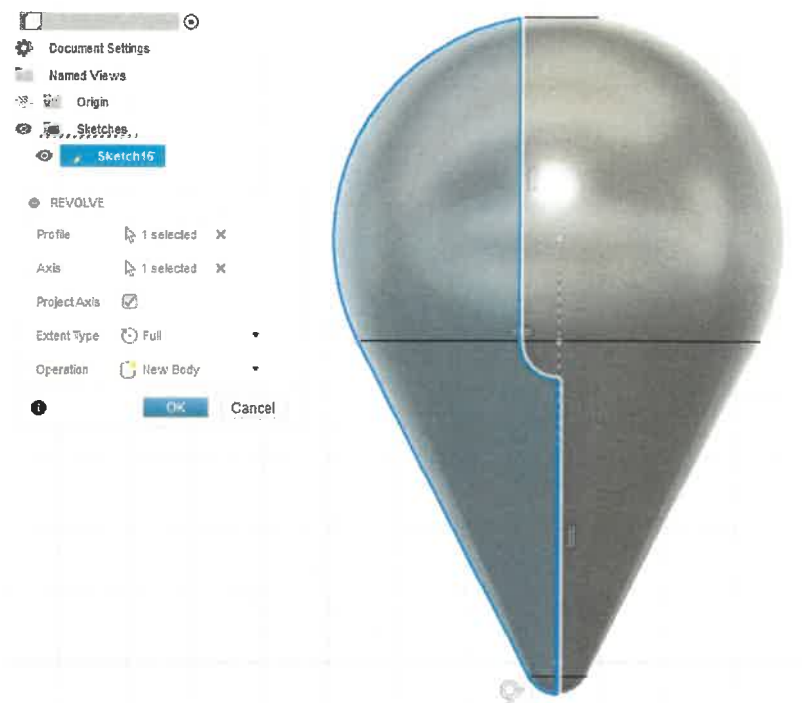


Select the 'Trim' option to remove unnecessary lines from both the slot and stand.

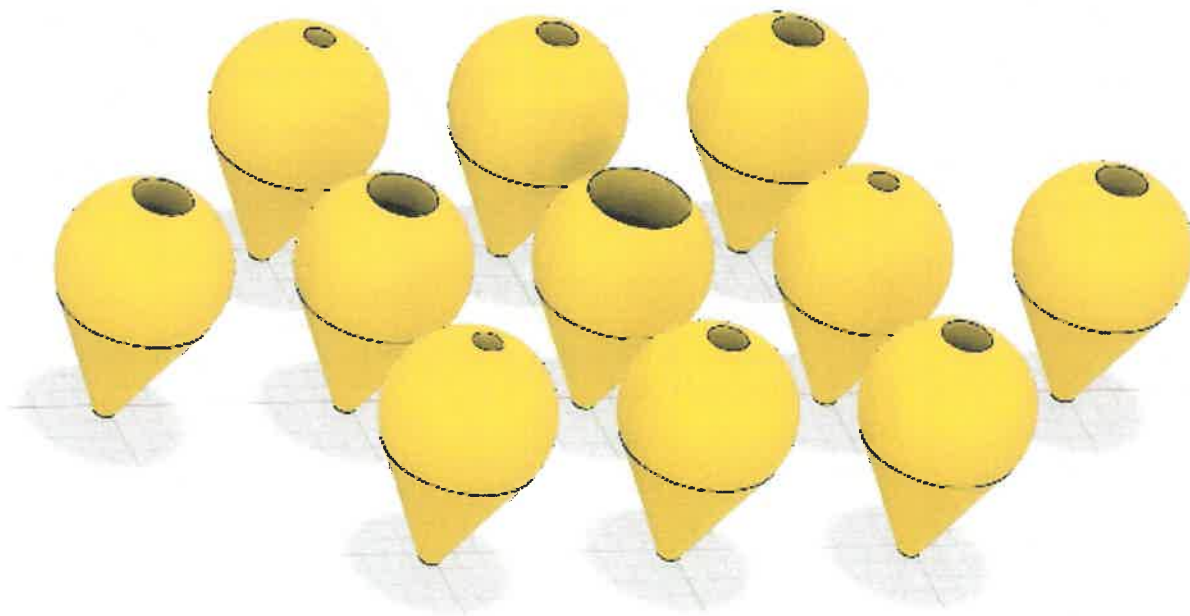


Once the sketches appear to be done, slice the stands vertically in two equal halves and delete their second halves.

This can be done by selecting the '2-Point Rectangle', clicking on the lowest point of the stand and clicking somewhere that the rectangle can overlap the entirety of half the stand.

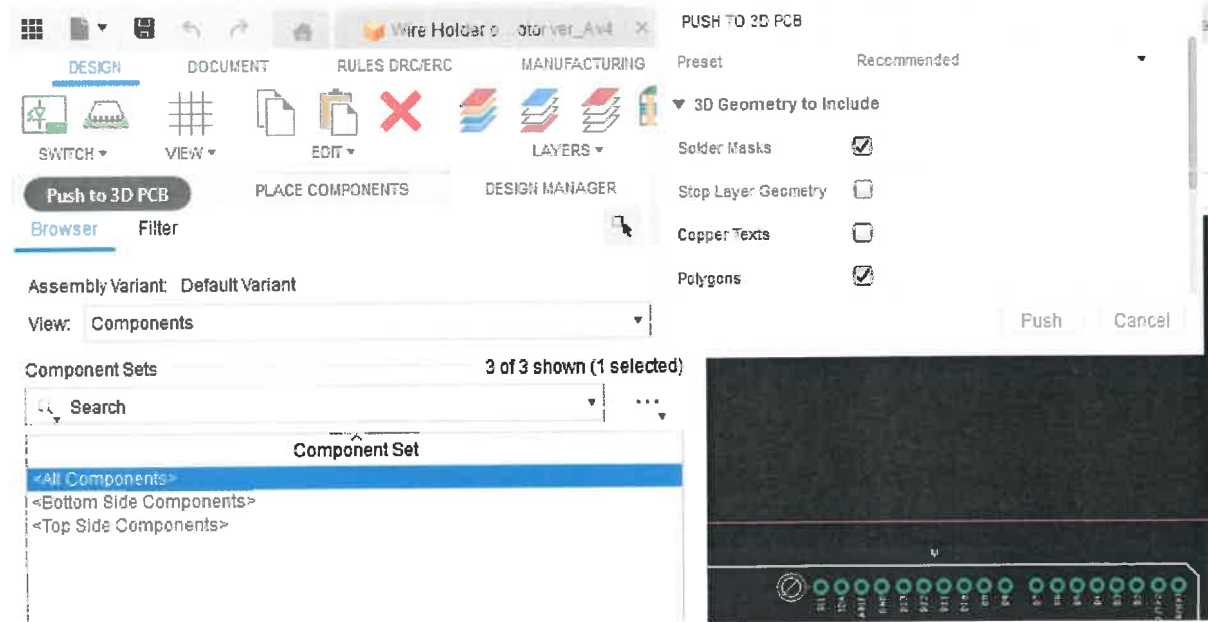


Select 'Revolve' in the 'Create' drop-down menu and select the stand half as the 'Profile'. Select the centermost vertical line as the 'Axis', and it should develop a preview of the 3D shape. Select 'Extent Type' to 'Full' and click 'OK' to produce the 3D shape. Repeat for other sketched halves.



Here are a group of examples of the 3D wire stands made of yellow plastic.

1.1.7 How to Push a Circuit Design into 3D PCB



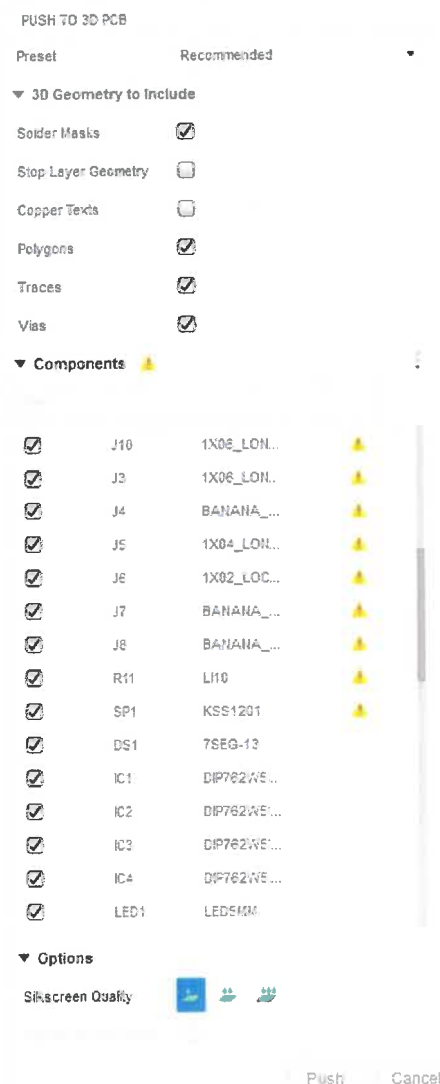
To produce a 3D render of the circuit board, go to the 'Design' tab and select 'Push to 3D PCB'.

Before a 3D rendering is produced, 3D Geometry options, a list of components, silkscreen quality and the choice to replace board outline are provided.

3D Geometry options are checkboxes to decide whether the listed features can be rendered. Components display respective designations, codes and whether they are given a non-cuboid 3D design for visual accuracy in simulation.

Silkscreen quality determines how much rendering to the silkscreen should it simulate while using up more computer processing in correlation.

Manually in the project, preset is set to 'Recommended' and 'Copper Texts' are ticked.

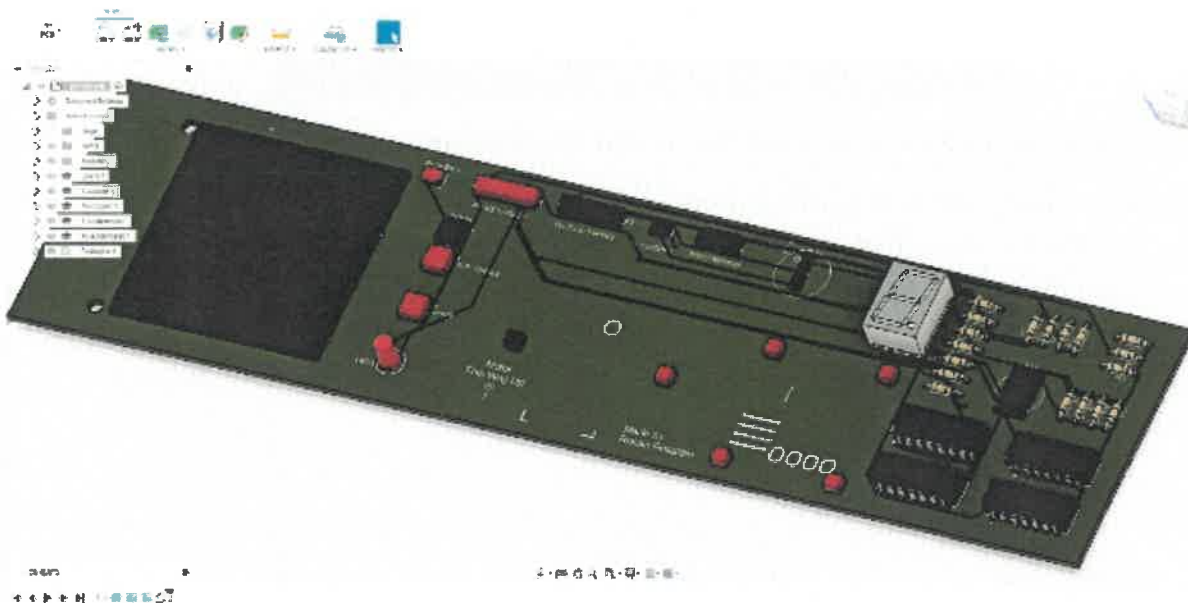


1.1.8 How to Place Text on Circuit Design



Top-left: Layer 1 is not recommended to apply text to the circuit board, otherwise it could end up producing an exposed copper etching of text on the 'Top' side of the PCB.

Top-right: Layer 25 will produce white sketching of text for visual indication or such purposes.



While it will not ideally look like the final design, the cuboid placeholder shapes must appear as such due to no 3D design being supplied to their respective components.

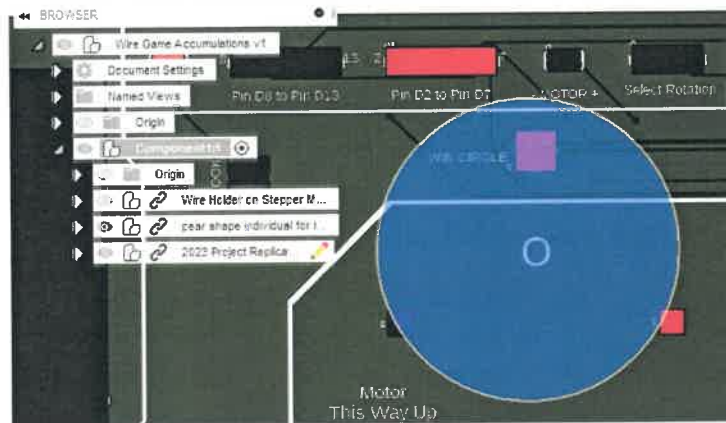
See, for example, the pin connector ports on the top of the PCB; banana connectors and Arduino to the left; drill holes in the middle; and the potentiometer surrounded by what clearly represents resistors.

Later steps in progress require acquisition of 3D printing and CAD (computer-aided design) software.

1.1.9 How to Insert Components to a New Project

While in a new and saved project file, there is the ability to show Data Panel (icon of 9 squares) and show a folder of components needed to assemble.

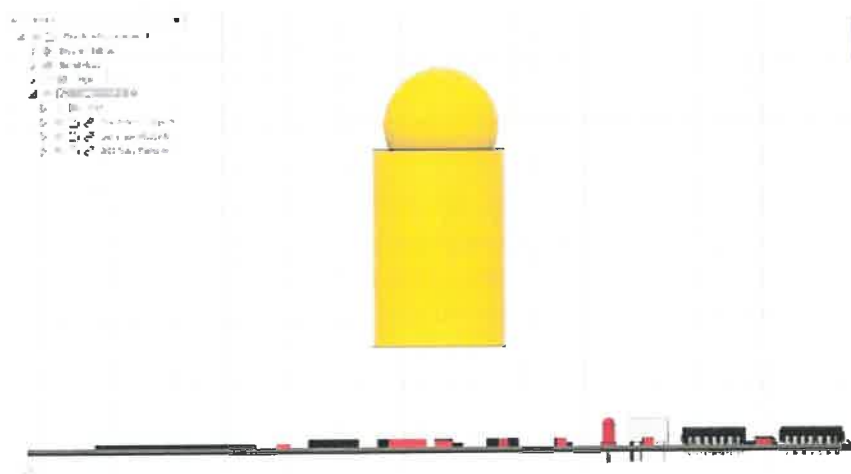
In the displayed folder contents, right-click on a component and select 'Insert into Current Design'. Repeat with other relevant components and individually drag them where appropriate.



The object highlighted in blue (large circle) is the wire stand holder.

The assembling mode of development can be a visual reminder for consequences of wrongly-placed components on the PCB and other user-made components. Here, remodification starts at Circuit Design.

The 28BYJ-48 component is ~3cm in height, including its external turning node. Stacking it with the user-produced wire holding components will produce the overall product into the following configuration (scaled down):

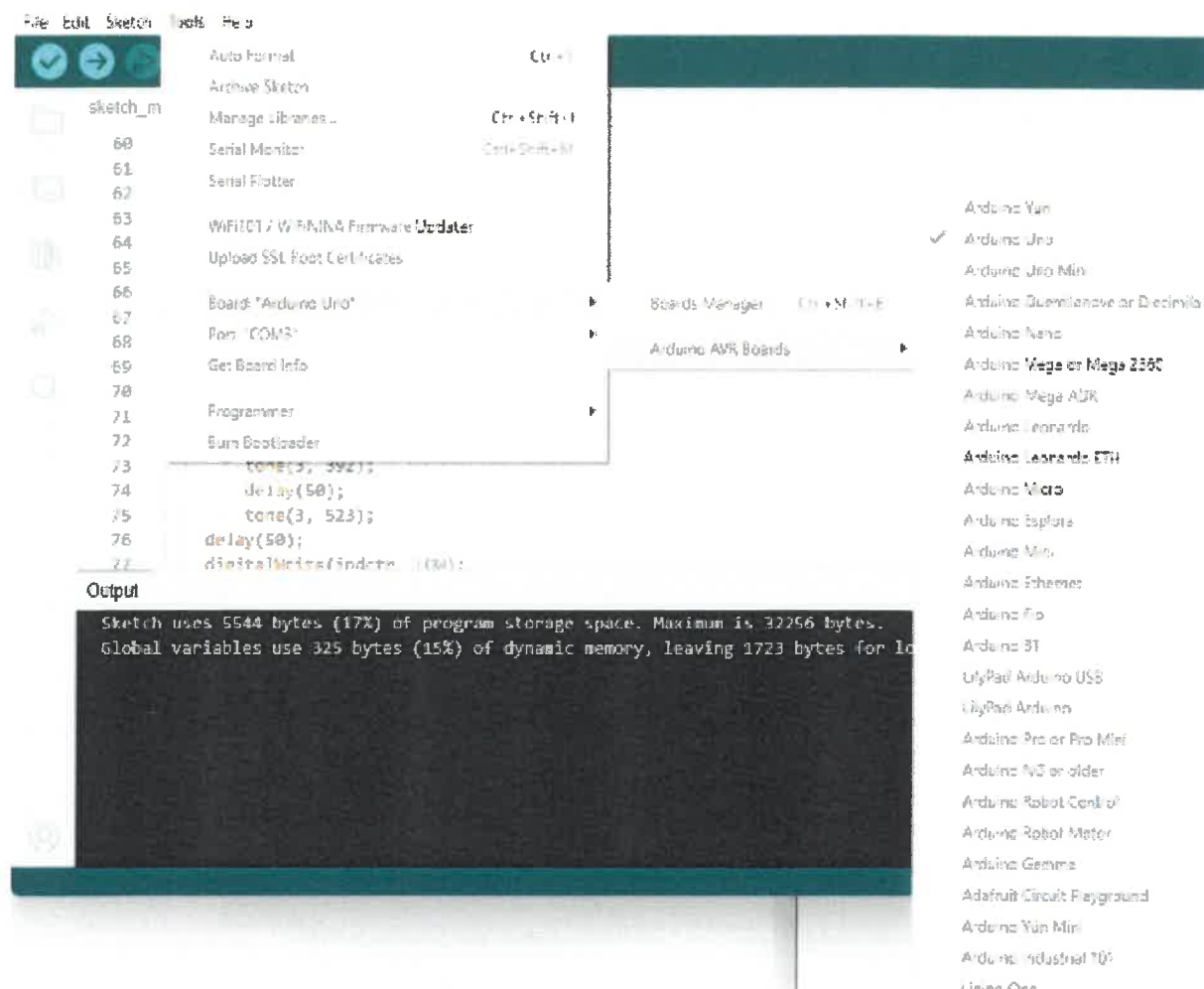


And because the hook rod is ~10cm, the hook rod is considered not to be assembled in case of design complications.

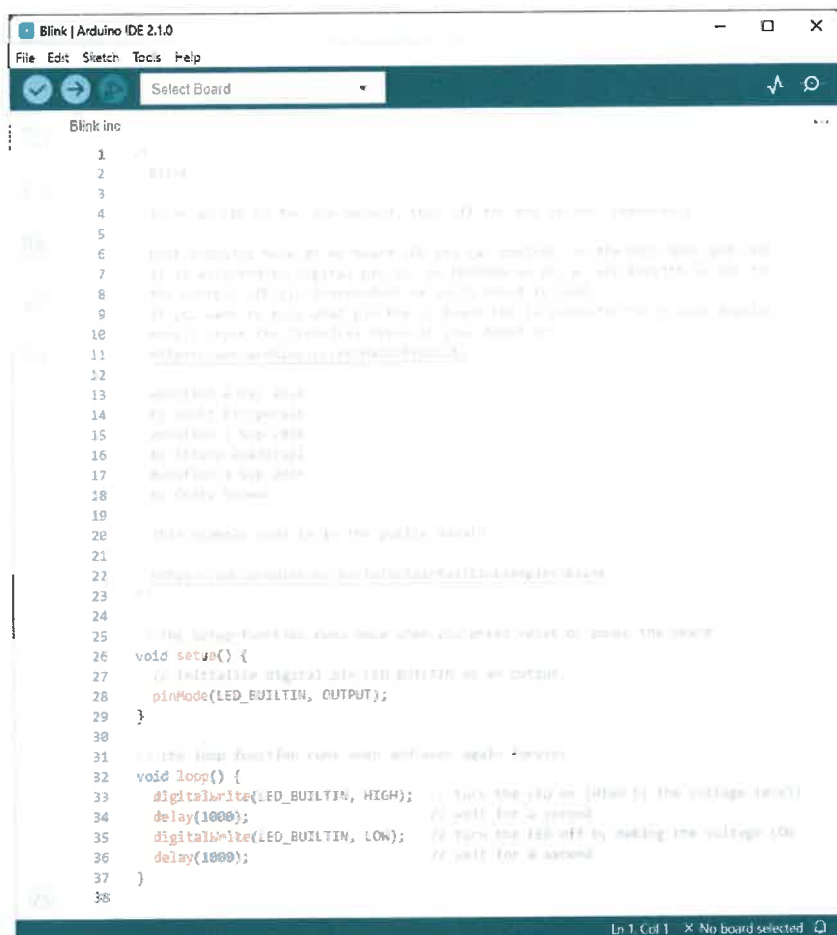
1.2 Arduino IDE

1.2.1 How to Demonstrate a Basic 'Blink' program on Arduino UNO using Arduino IDE

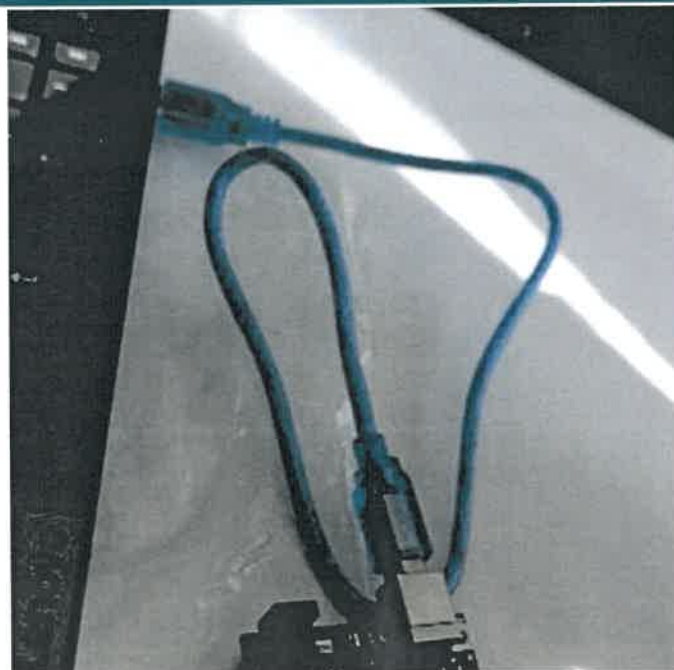
Once Arduino IDE is downloaded, make sure to give it access to device features in case code uploading ends up improper.



Test the Arduino by connecting it with the provided Male-Male USB-A to USB-B cable into the programming device. And open an example code built-in via. 'File' drop-down, hover over 'Examples' to '01.Basics' and click on 'Blink'. Click on the right arrow icon to upload the program to the Arduino. (Click on the tick icon if verification of code is chosen.)



```
1 //
2 // Blink
3 //
4 // This example code is for the Uno board. It will turn the LED on and off.
5 //
6 // The LED is connected to digital pin 13. The LED is connected to the GND pin.
7 // The LED is connected to the GND pin. The LED is connected to the GND pin.
8 // The LED is connected to the GND pin. The LED is connected to the GND pin.
9 // The LED is connected to the GND pin. The LED is connected to the GND pin.
10 // The LED is connected to the GND pin. The LED is connected to the GND pin.
11 // The LED is connected to the GND pin. The LED is connected to the GND pin.
12 //
13 // The LED is connected to the GND pin. The LED is connected to the GND pin.
14 // The LED is connected to the GND pin. The LED is connected to the GND pin.
15 // The LED is connected to the GND pin. The LED is connected to the GND pin.
16 // The LED is connected to the GND pin. The LED is connected to the GND pin.
17 // The LED is connected to the GND pin. The LED is connected to the GND pin.
18 // The LED is connected to the GND pin. The LED is connected to the GND pin.
19 //
20 // This example code is for the Uno board.
21 //
22 // The LED is connected to the GND pin. The LED is connected to the GND pin.
23 //
24 //
25 // The LED is connected to the GND pin. The LED is connected to the GND pin.
26 void setup() {
27   // Initialize digital pin 13 as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30 //
31 // The loop function runs over and over again forever.
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // Turn the LED on (HIGH is the voltage level)
34   delay(1000); // Wait for a second
35   digitalWrite(LED_BUILTIN, LOW); // Turn the LED off by making the voltage LOW
36   delay(1000); // Wait for a second
37 }
38
```



The 'TX' and 'RX' lights will light up for a short duration then turn off. The 'L' light will continuously blink.



1.2.2 How to transfer TinkerCAD code to Arduino UNO

The code that was developed in TinkerCAD should then be copy-pasted to Arduino IDE.

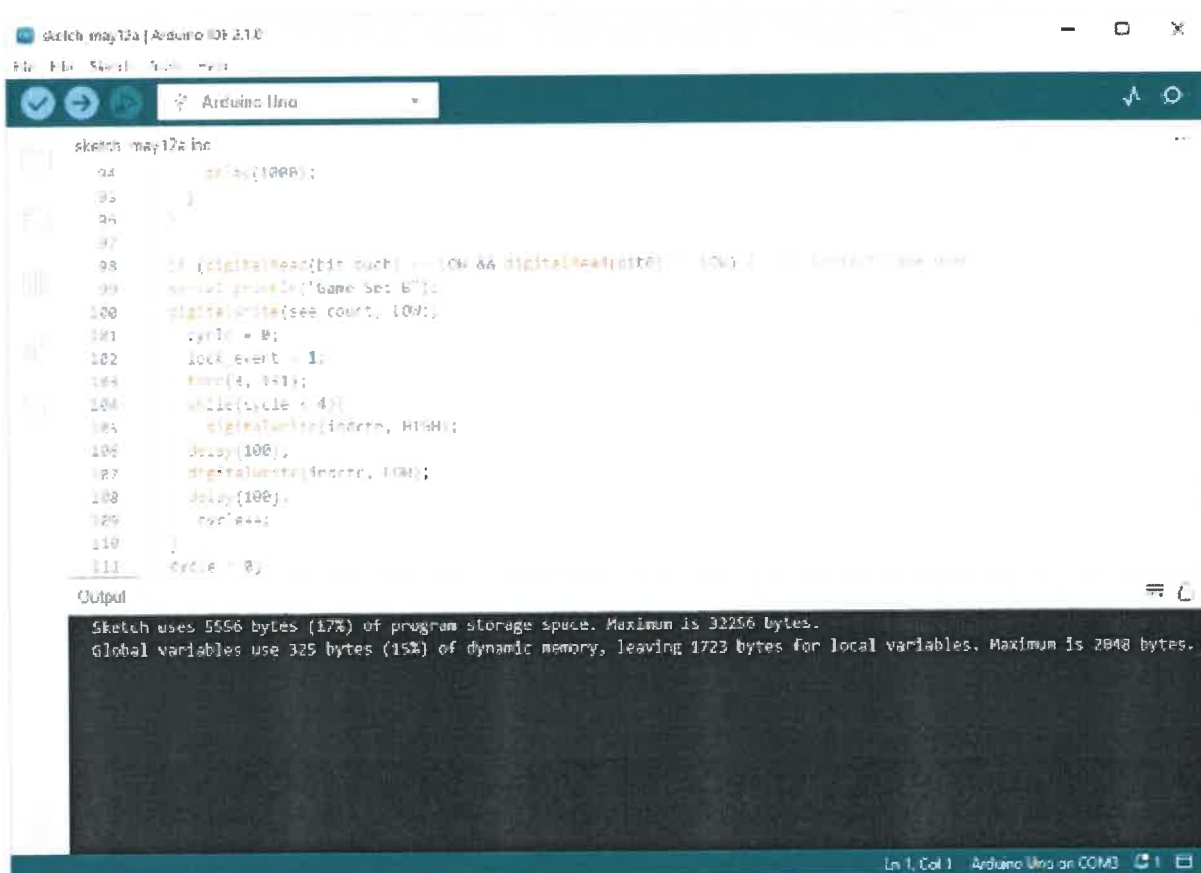
Go to the 'Tools' drop-down and select corresponding options for the Arduino UNO.

Go to the TinkerCAD project code and press 'Ctrl' and 'A' or right-click and select 'Select All'.

Right-click and select 'Copy' or press 'Ctrl' and 'C'.

Go to Arduino IDE. Erase pre-existing code, then paste via. right-click and select 'Paste' or press 'Ctrl' and 'V'.

In TinkerCAD, the code itself is partially guaranteed as a finished development during testing that it is now transferred to Arduino IDE. The code can be edited since transferral of code should not be interferable through typing in the middle of the process.



The screenshot shows the Arduino IDE interface. The top toolbar includes icons for opening files, saving, compiling, uploading, and monitoring. The 'Tools' menu is open, showing 'Arduino Uno' selected. The main editor displays the code for 'sketch_may12a.ino'. The code includes comments and functions for setting up a game set, writing to a display, and controlling a motor. The output window at the bottom shows the memory usage of the sketch.

```
sketch_may12a.ino
94  digitalWrite(1000);
95  }
96
97
98  // (digitalRead(bit Buch) == LOW && digitalRead(bit2) == LOW) // Contact Close (LOW)
99  serial.println("Game Set: B");
100  digitalWrite(see count, LOW);
101  cycle = 0;
102  lock_event = 1;
103  turn(4, 44);
104  while(cycle < 4){
105    digitalWrite(inderr, HIGH);
106    delay(100);
107    digitalWrite(inderr, LOW);
108    delay(100);
109    cycle++;
110  }
111  cycle = 0;
```

Output

Sketch uses 5556 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 325 bytes (15%) of dynamic memory, leaving 1723 bytes for local variables. Maximum is 2048 bytes.

Ln 1, Col 1 Arduino Uno on COM3

2. Components in Use of Product

2.1 Initials in Integrated Circuits

74xx Series

LS

Integrated circuits that contain LS on their labels stand for '*Low-power Schottky*' and comprise of *Transistor-Transistor Logic* (TTL) circuitry. 74LS ICs sink up to 16mA but source up to 2mA. Additionally, the entire 74xx family are often called the 'TTL series' despite later builds not using TTL circuitry.

"Schottky"?

In terms of 'Schottky', Schottky diodes are a type of semiconductor diode having a low forward voltage drop ($\sim 0.4\text{V}$, typically ± 0.1) and very fast switching speed. Schottky diodes can be used in a variety of wave shaping, switching, and rectification applications like other diodes; but more renowned for its lower voltage drop parameter due to being made up of a metal electrode bonded to an N-type semiconductor instead of a P-type material taking place. (ElectronicsTutorials, c. 2019)

HC or HCT

74HCxx ICs e.g: 74HC00, 74HC08, 74HC93, etc.

The HC or HCT initials in the selected examples indicate that the ICs use *Complimentary Metal-Oxide Semiconductor* (CMOS) transistors of *high-speed* varieties. 74HC or 74HCTs can sink up to 20mA or source up to 4mA.

HCTs specifically are available to be combined with LS ICs, but a drawback to HCTs (albeit minor) involve their lower immunity to noise.

Overall

All mentioned sub-families (except for 74HCs, 2 to 6V, tolerating small fluctuations) require that a smooth, regulated 5-volt supply ($\pm 0.25\text{V}$) is to power circuits involving them. The drawback from enforcing this parameter is that battery operation will not be recommended for common structure. But single-board microcontrollers such as the 'Arduino Uno', etc. can function consistently with the 5-volt parameter, allowing convenient use with such ICs.

Said families have a gate propagation time of 10ns for a signal to travel through a gate. (Electronics Club, n.d. a)

4000 Series

CD

The 'CD' designation indicates pin compatibility with the older 4000 series CMOS logic family. The pin outs were intentionally made different to further distinguish between CMOS (4000) and TTL (74xx) ICs. (cabraham, 2016)

Examples of CD ICs include: 'CD4511', 'CD4017', 'CD40109', etc.

Such ICs have a supply parameter between 3 to 15 volts, with tolerance to small fluctuations. And their gate propagation time takes typically 30ns by a 9-volt supply, whereas lower voltages take longer. (Electronics Club, n.d. b)

For both series, you can connect a transistor to sink further current from their outputs.

There can be ICs such as the 7493 that may require output regulation through capacitors to prevent noise interference from display unintended results. (Texas Instruments, 2022)

Texas Instruments - Important Notice and Disclaimer

The following image specifies that use of Texas Instruments products will and must be mandatory (assuring any extent is allowed) in this project.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale or other applicable terms available either on [ti.com](https://www.ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2022, Texas Instruments Incorporated

Image of Important Notice and Disclaimer by Texas Instruments (Semiconductor, 2003a)

2.2 Integrated Circuits (ICs)

2.2.1 4511 ICs – 4-Bit BCD-to-7 Segment Latch Decoder Drivers

4-Bit BCD-to-7 Segment Latch Decoder Drivers are a sort of logic Integrated Circuit (IC) and acts as sourcing for 7-Segment displays when an appropriate IC (e.g: 74HC93) and other connections are linked appropriately to provide respective combinations of output.

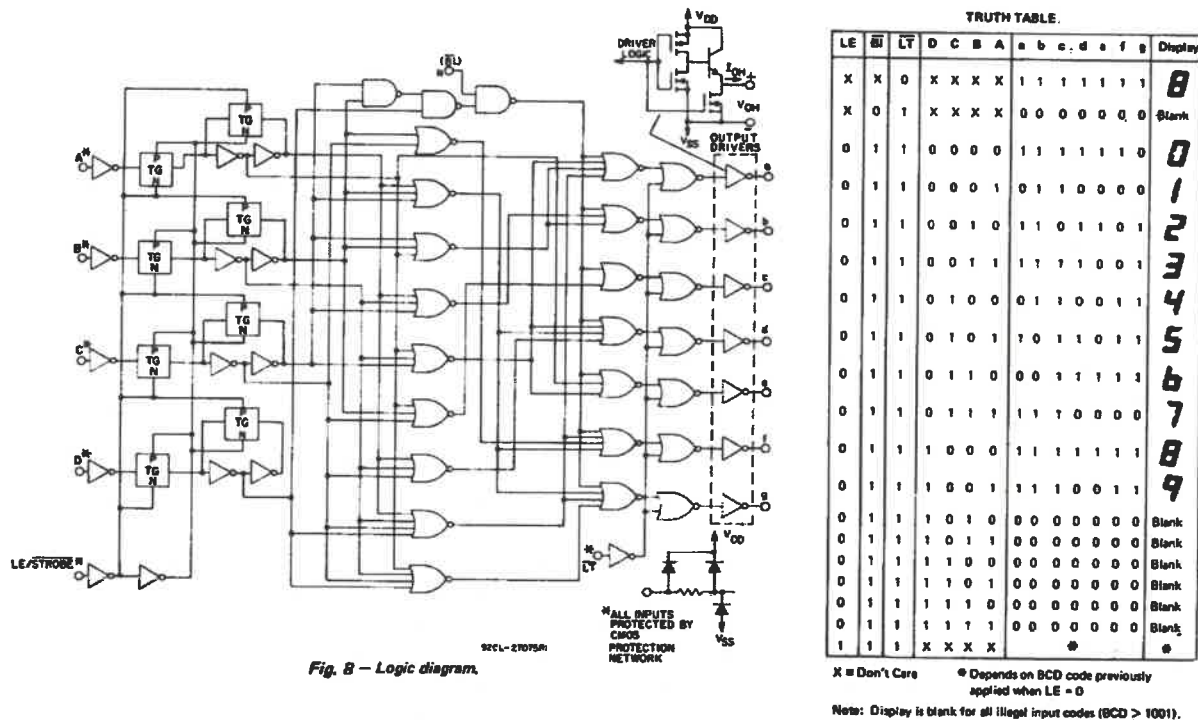


Image of CD4511B Logic Diagram + Truth Table (Semiconductor, 2003b)

If *Lamp Test* (LT) is set to a LOW state, the IC lights up all seven outputting sources (preferably and for example, segments) regardless of other inputs.

If LT is HIGH and *Blanking* (BI) is LOW, all segments will not light up no matter the states of the 4-Bit inputs and *Latch Enable* (LE).

If both LT and BI is HIGH as well as LE, the state of the 7-segment display will persist at its last state before LE is set to a HIGH state.

But if LE is LOW while LT and BI is HIGH, the segments will light up depending on the state of the pins connected to the segments.

Latch Enable gives a suggestion that “when the wire game ends”, the state of the 7-segment displays will “freeze” to provide a final time and mistake count.

Transmission Gates (TG) (see left of image) determine whether inputs A to D will matter in changing output states. If LE (or any name of control rail) is LOW, the columns of TGs' outputs will remain unchangeable or of immovable quantity of impedance. Their negative rails will be HIGH and their positive rails will be LOW, concluding that a logical nullification of any further input state change happens until LE is HIGH. (Pavankalyan, n.d.) (Basic Electronics Tutorials, n.d.)

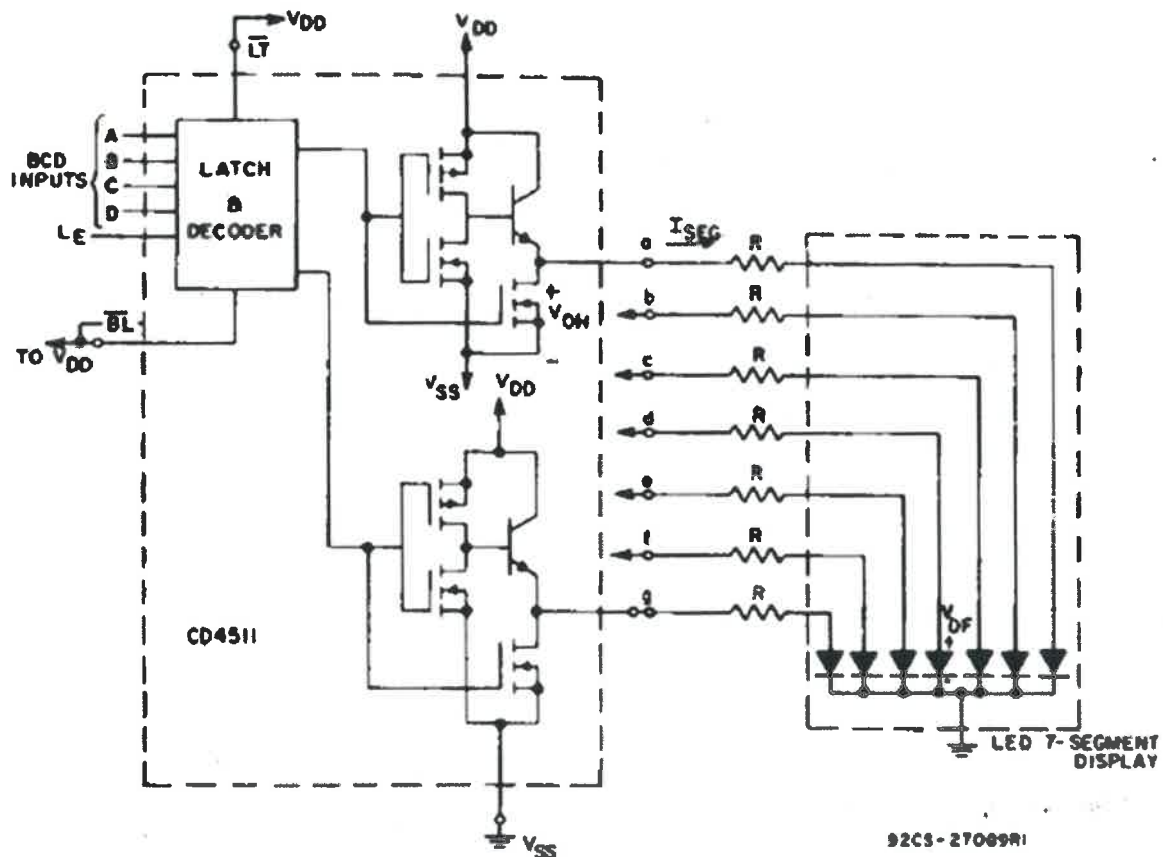
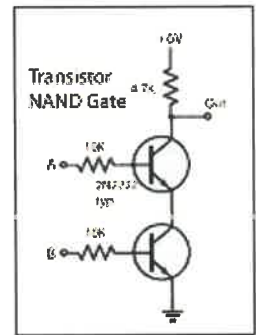
The vast quantity of *inverter gates* (triangles with a dot on one end) are so that they provide a miniscule buffer (often in nanoseconds) for processing input changes and to negate the input to an opposite output.

2.2.2 7400 ICs – Quad 2-input NAND Gate

A 7400 IC is seen to be made up of four NAND gates with individual source and ground pins for appropriate voltage reference. Pin 14 generally acts as a voltage supply pin and 7 acts as a ground rail for the IC.

Image of 4511B Logic Diagram + Truth Table (Nave, n.d.)

An example transistor NAND gate comprise of two transistors e.g: two 2N2222s, resistors for each base pin for those transistors, voltage supply, input voltages (preferably digital signals), a ground rail, and an output rail.



Duty Cycle = 100%

$I_{SEG} = I_{DIODE_{AVG.}} = 20 \text{ mA at Luminous Intensity/Segment} = 250 \text{ microcandles}$

$$R = \frac{V_{OH} - V_{DF}}{I_{SEG}}$$

Fig. 15 - Driving common-cathode 7-segment LED displays (example Hewlett-Packard 5082-7740).

(Semiconductor, 2003c)

The inclusion of enhancement-mode (arrow pointing left is negative, and vice versa) metal-oxide-semiconductor field-effect transistors (MOSFETs, not BJTs) effectively regulates the current flowing into each output pin (from A to G). (LearningaboutElectronics, c. 2018)

2.2.3 7493 ICs – 4-bit Binary Counter

The 7493 IC works as a 4-Bit Ripple Counter.

Figure 7-1 shows three blocks involved in a 7493 IC: NAND Gate Memory Reset, an individual D-type flip-flop, and three consecutive D-type flip-flops.

The “shared control logic” indicates that both ‘Memory Reset’ rails should engage with LOW signals each to activate the reset of the entire IC.

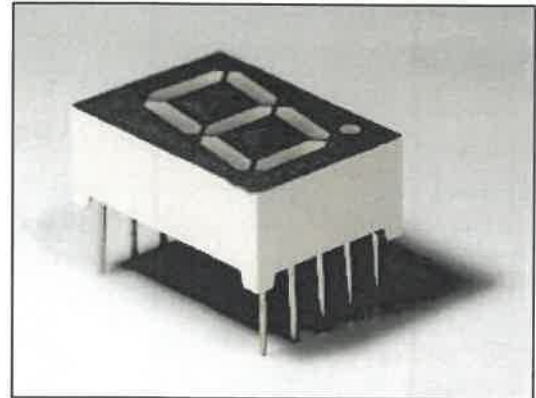
2.3 Other Components

2.3.1. 7-Segment Display

7-segment displays are designed with character readability as a common intended use in electronic devices.

They have ten pins which function for the following: eight LEDs that represent a “segment”, including the decimal point; and two common (both anode or cathode) pins (often central to the other pins) for providing either power or ground rail for the LEDs to correctly glow.

Figure 1: 7-Segment Display. (Halasz, 2007) Photograph by Peter Halasz, distributed under a CC BY-SA 3.0 license.



Resistors must be connected to all used segment pins (excluding common pins), otherwise too much current will flow in those without it, causing damage to the component.

As illustrated in Figure 2, pins 7, 6, 4, 2, 1, 9, 10 and 5 (A to G, then Decimal Point) light up respective segments. The mentioned pins must require a current-limiting component e.g: resistor, so to not overpower the segments.

Common pins, 3 and 8, are suggested to act either as a ground rail i.e: ‘Common Cathode’, or voltage supply i.e: ‘Common Anode’ for all segments.

For example: if the 7-segment display is a ‘Common Anode’ type, lighting up segment ‘C’ requires a 1kΩ resistor connected to pin 4 and to a ground rail while the display’s common pin sources voltage.

For ‘Common Cathode’ displays, a resistor connects to a segment pin and a voltage source supplies through it. Then the display’s common pin must connect to a ground rail.

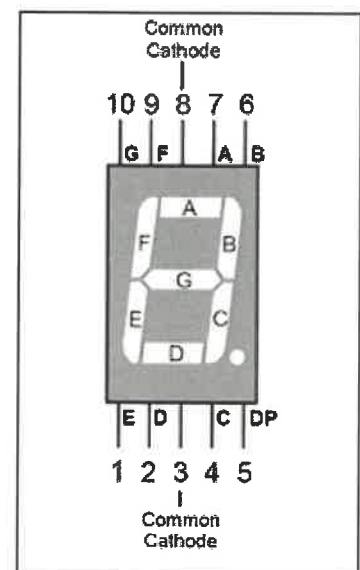


Figure 2: 7-Segment Display, Common Cathode. (WatElectronics, 2022)

3. Tables involved with Project

3.1 Prototype's Connections per Component

3.1.1 CD4511 – BCD 4-Bit Decoder

CD4511 – BCD 4-Bit Decoder		7493 – 4-Bit Binary Counter		Common Cathode 7-Segment Display	Power, Ground, No Connection, or etc.?	
1	Input 3	9	Output Bit 2	n/a	n/a	
2	Input 2	8	Output Bit 1	n/a	n/a	
3	Lamp Test	n/a		n/a	Power (enable visual)	
4	Blanking	n/a		n/a	Power (enable incr.)	
5	Latch Enable	n/a		n/a	(future idea, time stop O/P)	[etc.]
6	Input 4	11	Output Bit 3	n/a	n/a	
7	Input 1	12	Output Bit 0	n/a	n/a	
8	Ground	n/a		n/a	Ground	
9	E	n/a		1	E	n/a
10	D	n/a		2	D	n/a
11	C	n/a		4	C	n/a
12	B	n/a		6	B	n/a
13	A	n/a		7	A	n/a
14	G	n/a		9	G	n/a
15	F	n/a		10	F	n/a
16	Power	n/a		n/a	Power	

3.1.2 7493 – 4-Bit Binary Counter

7493 – 4-Bit Binary Counter		CD4511 – BCD 4-Bit Decoder		Same as main table subject.		Power, Ground, No Connection, or etc.?	
1	Clock 1	n/a		12	Output Bit 0	n/a	
2	Reset 1	n/a		n/a		Power	V+
3	Reset 2	n/a		n/a		740x O/P	[etc.]
4	NC	n/a		n/a		No Connection	NC
5	Power	n/a		n/a		Power	V+
6	NC	n/a		n/a		No Connection	NC
7	NC	n/a		n/a		No Connection	NC
8	Output Bit 2	1	Input 3	n/a		n/a	
9	Output Bit 1	2	Input 2	n/a		740x I/P	[etc.]
10	Ground	n/a		n/a		Ground	GND
11	Output Bit 3	6	Input 4	n/a		740x I/P	[etc.]
12	Output Bit 0	7	Input 1	1	Clock 1	n/a	
13	NC	n/a		n/a		No Connection	NC
14	Clock 0	n/a		n/a		"Timer" (1 st digit)	[etc.]
						740x O/P (others)	[etc.]

3.2 Pin Tables in Varying Formats

3.2.1 7400 and 7408 Connections – NAND & AND Gates

Input Pins	1	2	4	5	9	10	12	13
Output Pins	3		6		8		11	
Power Pin	14							
Ground Pin	7							

3.2.2 Time Increment/Decrement and Contact/Mistake Increment

- **Inputs – 3, 6, 8, 11**
 - 7493 – Pin 9 (Output Bit 1)
 - 7493 – Pin 11 (Output Bit 3)
- **Outputs 1+2, 4+5, 9+10, 12+13**
 - 7493 – Pin 3 (Reset 2)

Both CD40193 and 74193 ICs in comparison include ‘BCD/Binary Preset’ rails, but the 74193 includes presettable inputs and other different function(s) including “Asynchronous Parallel Load Enable”. (Semiconductor, 2003) (Fairchild Semiconductor, 1999)

3.3 DIP Settings [Voided/Cancelled]

Timer Number

B/A	0	1
0	000	300
1	180	300

Rotation Direction of wire (C & D)

D/C	0	1
0	None	Clockwise
1	Ant. Clw.	RANDOM

Speed of Rotation (E & F)

F/E	0	1
0	None	Slow
1	Fast	RANDOM

Sound Setting (G)

0	1
Sound OFF	Sound ON

LED Setting... I believe this is a bad idea (H)

0	1
Display OFF	Display ON

Let us not implement this setting. It will be a terrible design choice.

Life Counter will be set to either 9 or 10, depending on how logical complexity is handled.

Timer Counting

B/A	0	1
0	Increment	Decrement
1	Decrement	Decrement

Rotation Involvement 1

D/C	0	1
0	OFF	x
1	x	x

Rotation Involvement 2

F/E	0	1
0	OFF	x
1	x	x

Rotation Involvement Total

R12/R11	D	1
0	OFF	OFF
1	OFF	ON

Timer Digit 1 Setting (leftmost)

B/A	0	1
0	0	3
1	1	0

Timer Digit 2 Setting (Middle)

B/A	0	1
0	0	0
1	8	6

Timer Digit 3 Setting (Rightmost)

B/A	0	1
0	0	0
1	0	0

Bit 1 & 2

Bit 3 & 4

Bit 5 & 6

Bit 7

Bit 8

CANCELLED as of 3rd May 2023

The timer is removed due to time constraints.

These pair of bits will be replaced with a hidden command for what rotation the wire will spin. Done via pre-game functions during start-up.

A potentiometer and resistor pair can control it all time.

As is described, a bad idea.

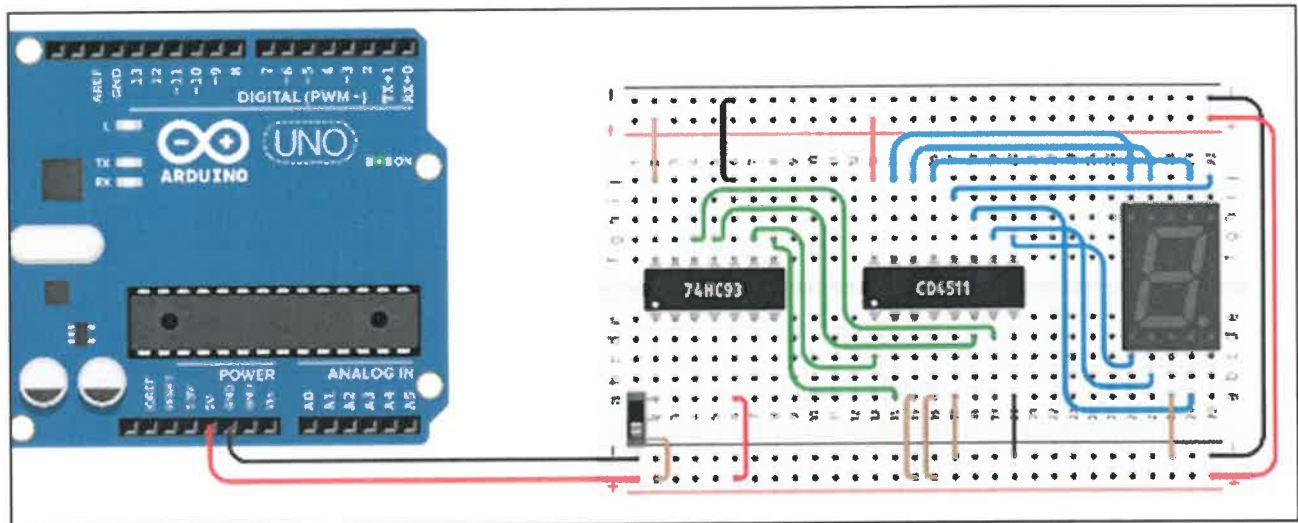
3.4 Pin Correspondence for Project

Variable Name	Pin Number on Arduino UNO	
	TinkerCAD & Final Release	Breadboard Prototype
Input		
bit0	2	7
bit_ouch	3	8
win_hit	4	12
Output		
indctr	6	4
attack	5	2
see_count	7	11
prelock	13	13
beepy	12	3
Step		
steA	11	10
steB	10	9
steC	9	6
steD	8	5

4. Non-Final Tests/Arrangements during Development

4.1 TinkerCAD - 4-Bit Segment Base-10 Display Test

Due to concerns of developmental capering, a 4-Bit 7-Segment Display Test will be performed.



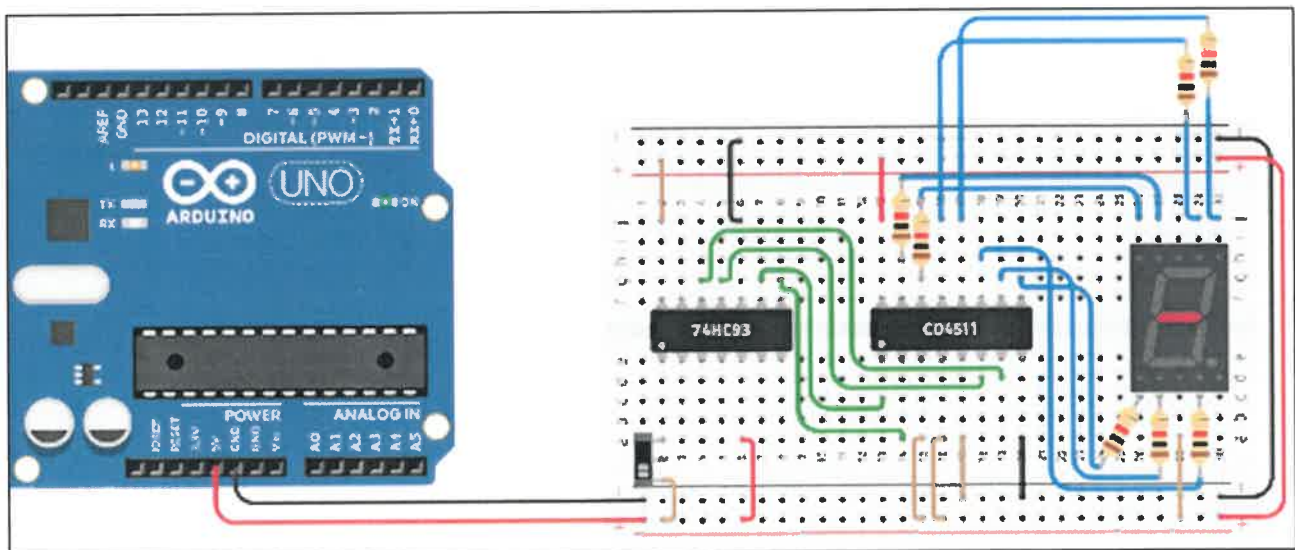
In the image above, red wires act as the source voltage rail. In this instance, 5 volts will be supplied to those connected.

Black wires act as ground rails for the voltage supply to loop back to its voltage source.

Green wires mostly connect to both ICs (74HC93 + CD4511) correspondingly whereas blue wires mostly connect from the CD4511 to the 7-segment display also correspondingly.

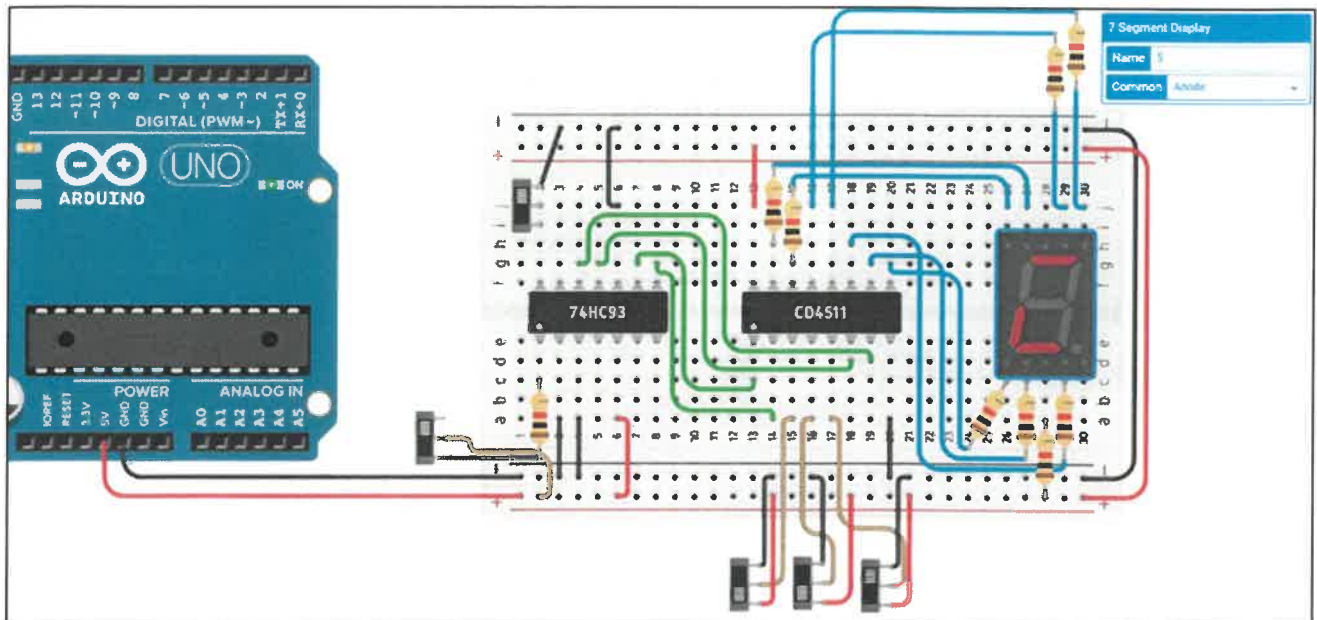
Brown wires are depicted as test rails due to uncertainty in testing permutations.

One reason why the given circuit above fails to work as intended is due to a lack of resistance across each blue wire. Too much current flows and the circuit fails to light up the 7-segment display.

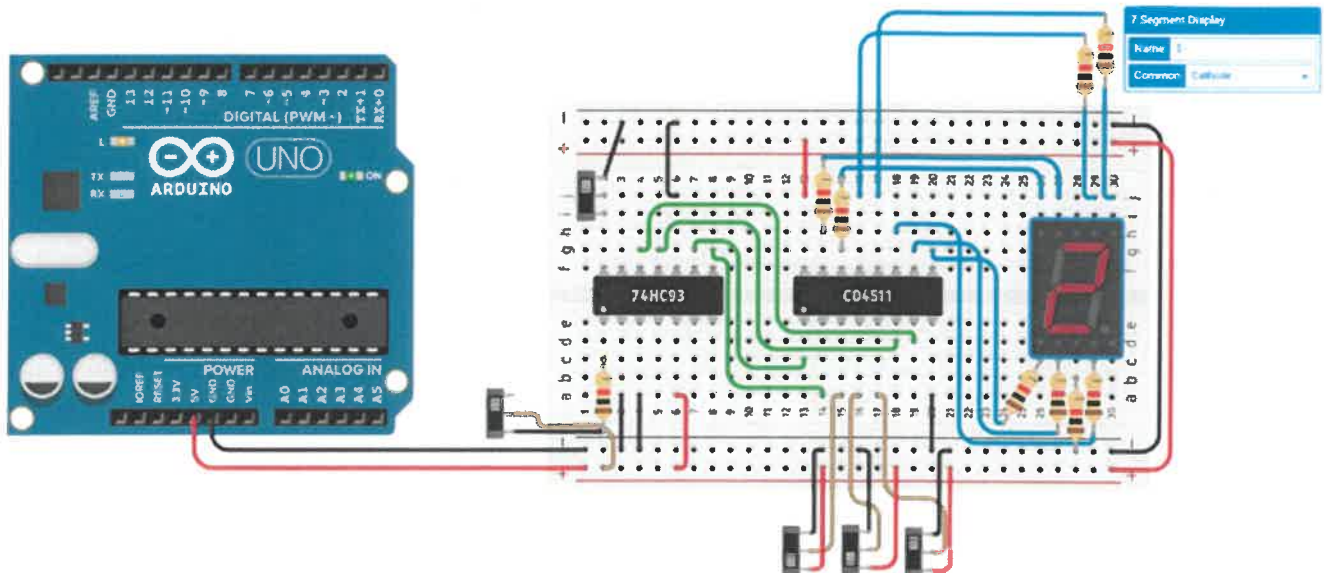


Despite adding resistors to the display's segment connections, the circuit above the current paragraph still fails to produce the intended result since it only lit up segment 'G' from the flawed wiring arrangement. However, as the slide switch was adjusted alternatively, it did count up but not lighting

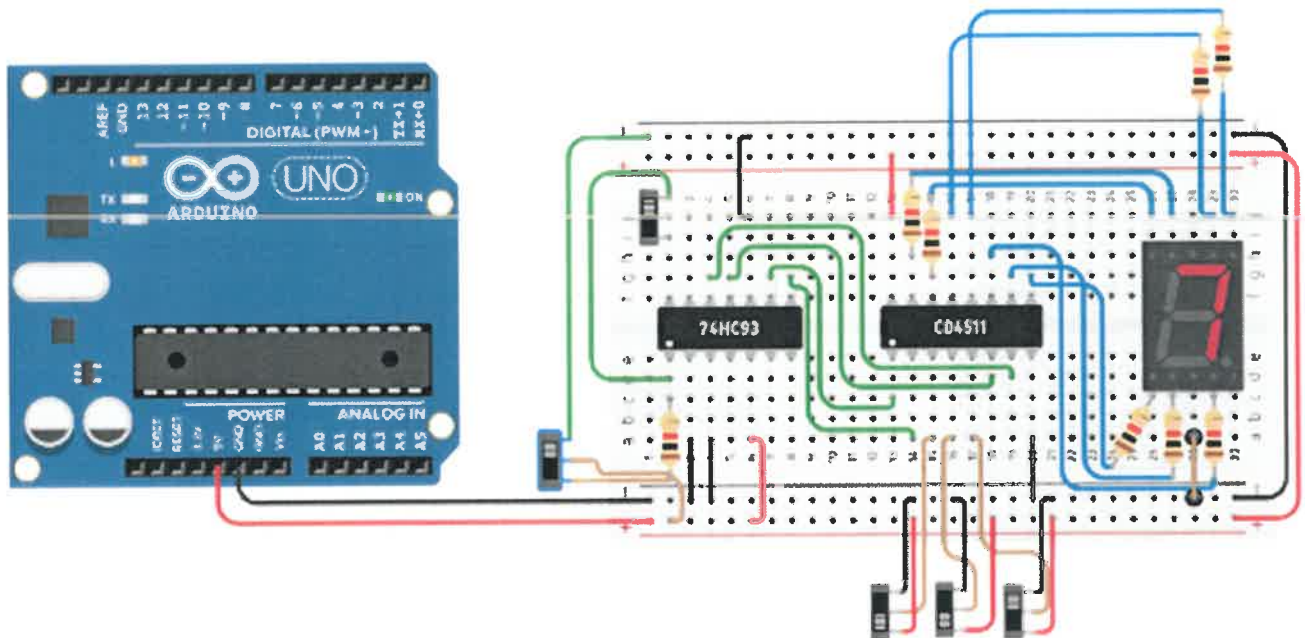
up the right segments. It had nothing to do with the reset pins (2 and 3) of the 74HC93 IC in this intent.



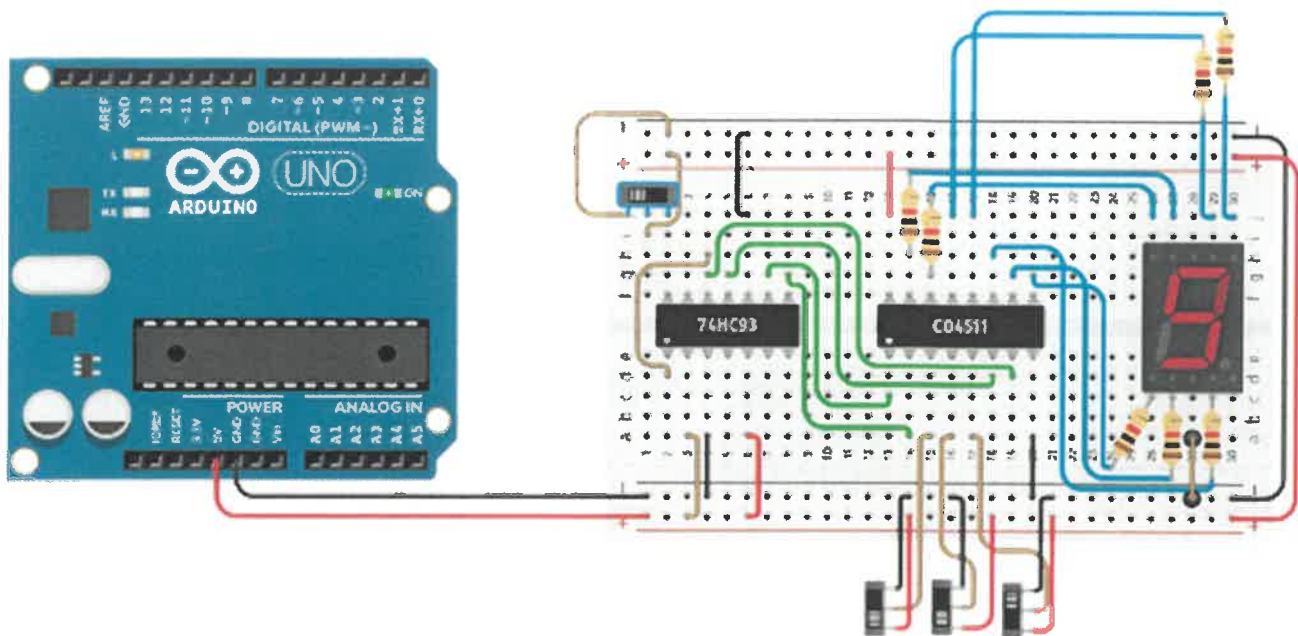
As indicated by the top-right window in the lower image, the 7-segment display's common setting was default to 'Common Anode' by the website (TinkerCAD). By the state of the 7-segment display, the number shown was meant to be the number four ('4') by the logic of the CD4511.



After changing the 7-segment display's 'Common' setting to 'Cathode', another problem arises as shown from the circuit above. Whenever the circuit turns on via. voltage supply, the number shown starts at the number two ('2').



However, when the circuit was counting upwards, either it would count two units per lower-left switch toggle and that the upper-left switch is toggled downward; or the circuit counts in an alternating sequence of 1 unit and 3 units if the upper-left switch is toggled upward.



By re-configuring the way in which the left half of the Ripple Counter/74HC93 IC is to be connected, the counter finally starts counting one unit per step-down toggle (as the leftmost slide switch slides left).

The re-configuration also allows the circuit to start the counter from zero ('0') every time it is powered ON.

What was altered between provided image evidence was that:

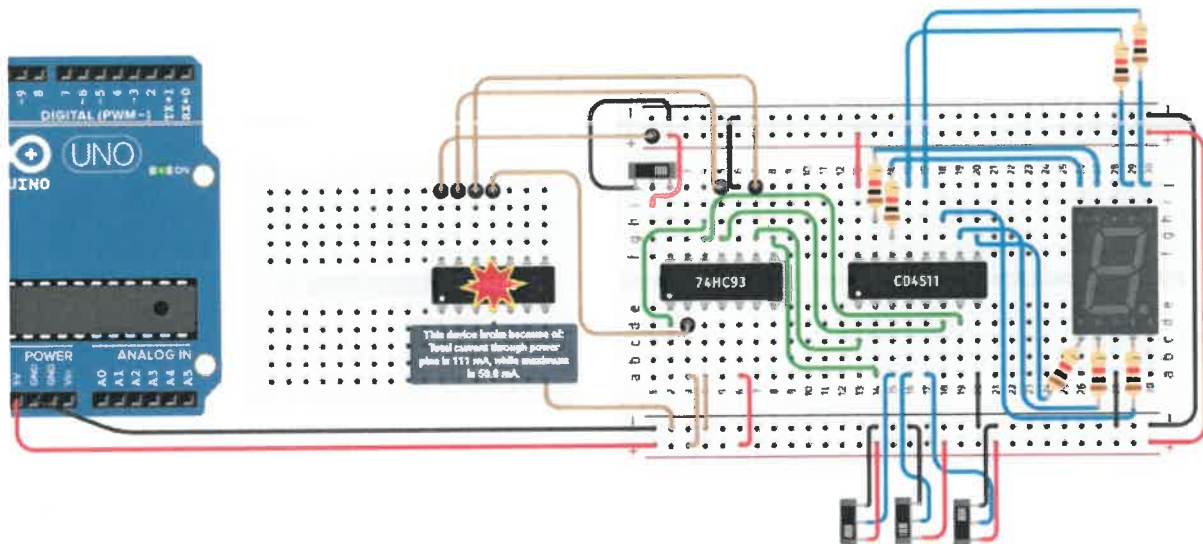
An excess slide switch and 1k resistor was removed (alongside the wires connected to them).

A former black wire from pin 2 of the 74HC93 IC is then connected to provide source voltage for aforementioned IC.

The remaining slide switch is supplied 5 volts to its common pin; its right pin connected to pin 14 of the 74HC93 IC; and the slide switch's left pin is connected to ground.

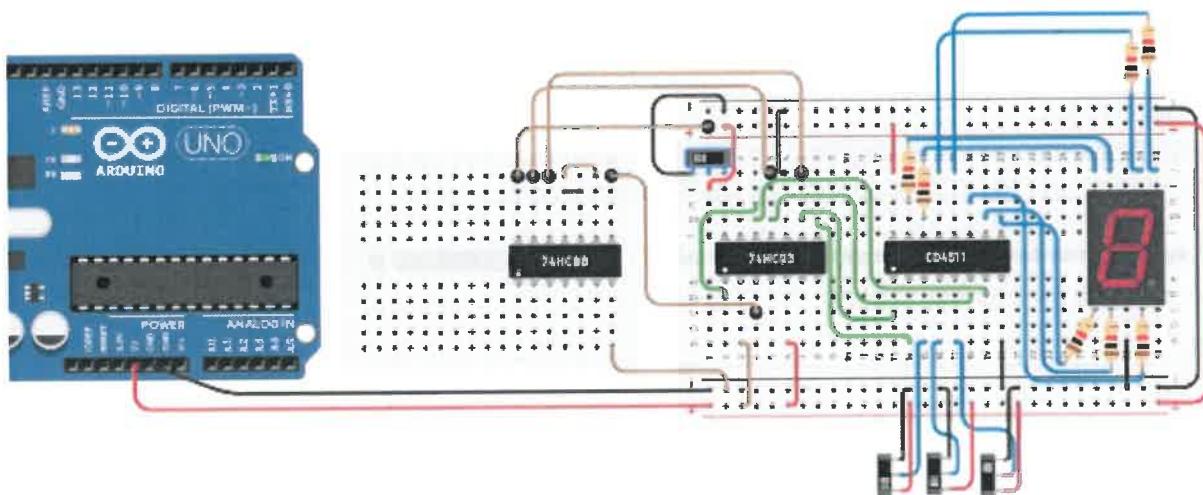
But no matter how the reset pins are configured, only setting them to HIGH states only will fully reset the state of the number to zero.

The next step is to apply a NAND gate by adding a NAND gate IC, namely a 74HC00 IC to the test circuit to perform a counting reset. An additional breadboard is to be supplied in the next step.



The above image evidence renders an immediate failure once the leftmost toggling counts past nine ('9'), resulting in the explosion of the 74HC00 IC.

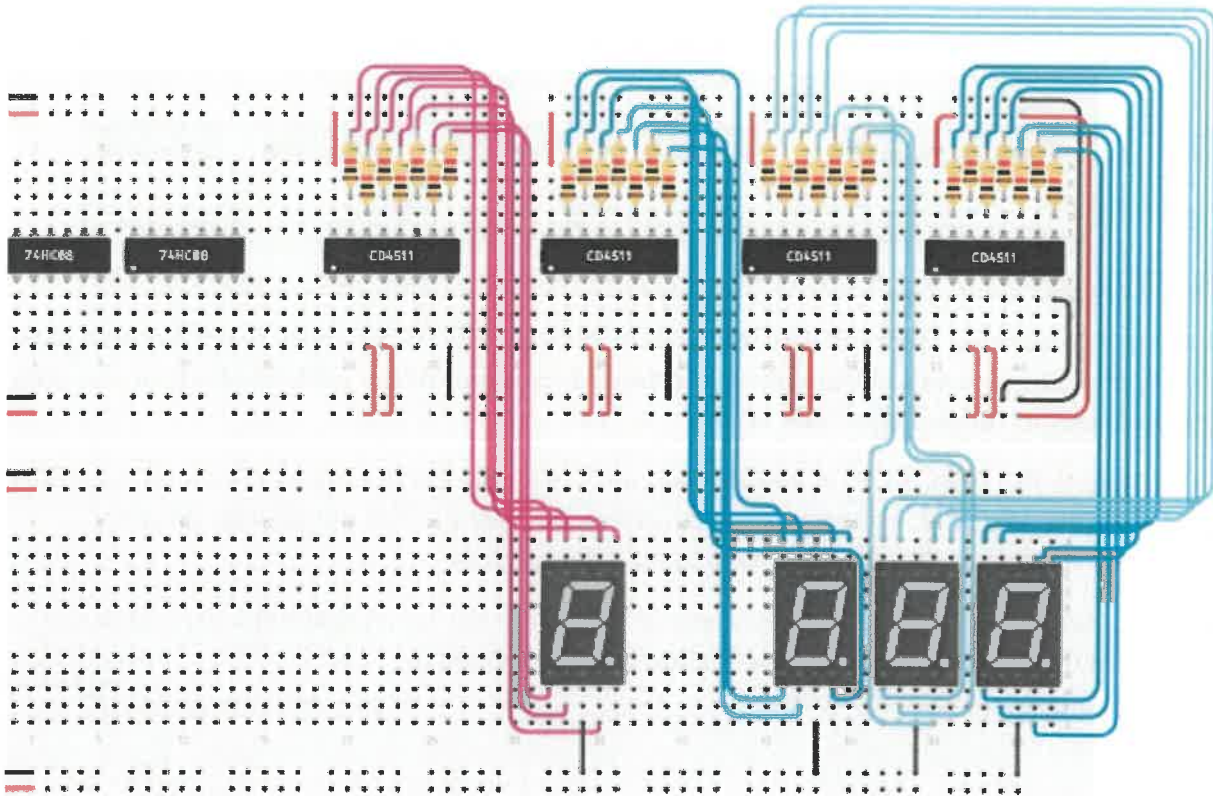
This was caused by an excess (brown) wire from pin 3 of the 74HC93 IC connecting to ground.



While the append may be perceived as wasteful in the future, two NAND gates were used so as to provide a step-up signal for pin 3 of the 74HC93 IC to reset its count back to zero.

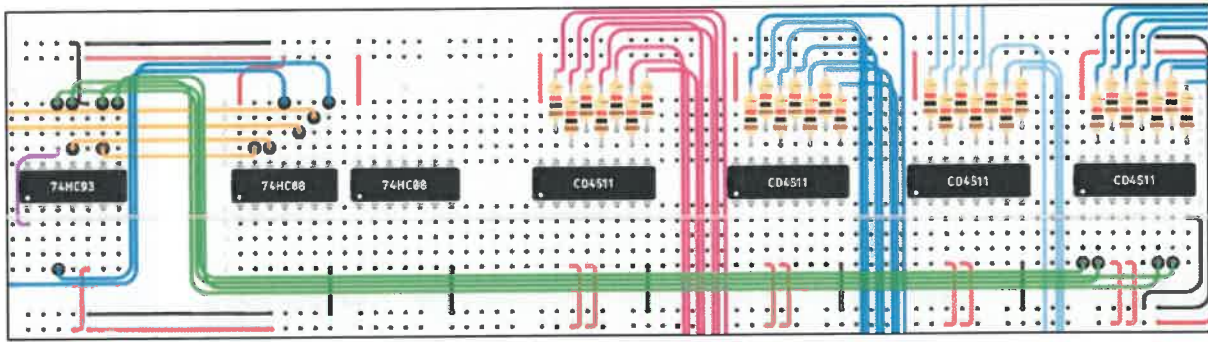
Pins 12 and 13 provide the inverted combining output for pin 11, whereas pin 11 connect to pins 10 and 9 that functions as another NAND gate for pin 8. Pin 8 will then connect to pin 3 of the 74HC93 IC.

4.2 TinkerCAD - Adding Three More 7-segment Displays



Starting off with a new canvas of breadboards (four default-sized horizontal breadboards, 2x2), three more 7-segment displays are to be lit up but with separate intents. The leftmost display is to be separated from the other three displays.

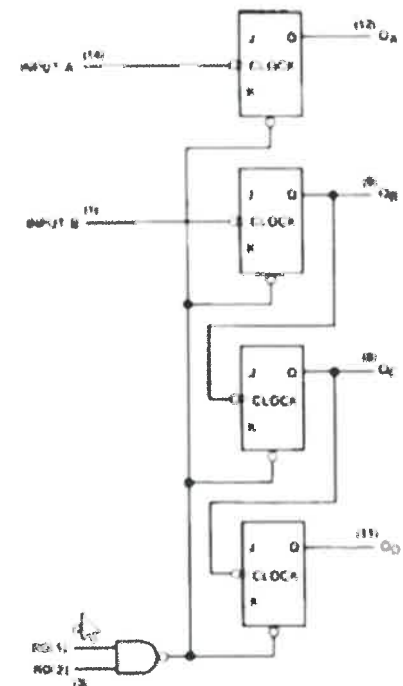
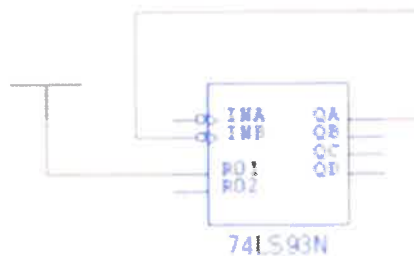
74HC08 ICs are to be the AND gates that connect to respective connections of the (not shown) 74HC93 ICs that make up the numeric (whether incrementing or decrementing) 4-bit signals.



The blue wires connecting from the used 74HC08 IC to two 74HC93 ICs mean that the output pins of the 74HC08 IC (pins 8 and 11) connect to the second reset pins of the 74HC93 ICs (pin 3 to both), ensuring the incrementation loops back to '0'.

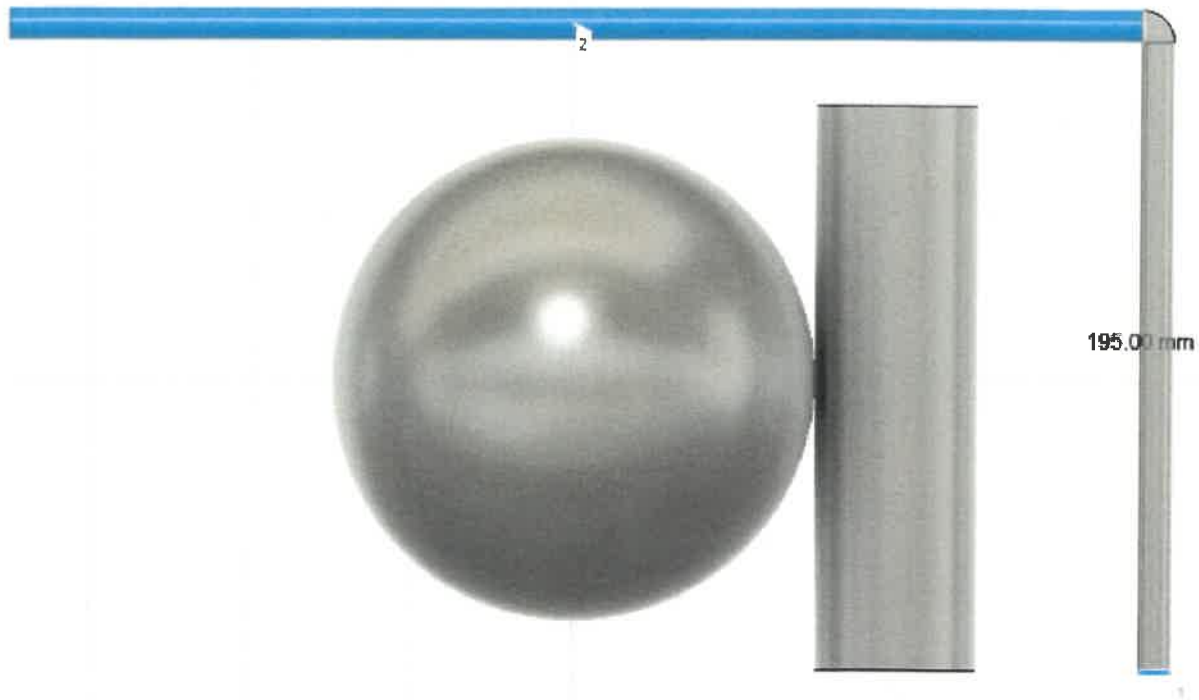
The purple wire to the 74HC93 IC connects from pin 12 (Output Bit 0) to pin 1 (Second Clock Pin). This is because the simplified schematic (see below) shows that the first output wire does not simultaneously connect to the second clock's input. (Haas, 2018)

Assumed conveniences of having two reset pins could be that the NAND gate-like behaviour is to cut down on configuration development and allow regulated input to process and minimise damage to the IC.



(Haas, 2018)

4.3 Fusion 360 - Marble and Metallic Contact Visualisation



Due to not acquiring another unnamed component – a 1.5cm – 4.0cm metal/steel ball, a flawed visualisation of a metallic contact (made with a silver tube bead and a paperclip) is barely developed without the ability to create tangential surrounding to the steel ball.

5. Code Batches – Full Code Variants used in Arduino UNO

5.1

```

1 // #include <Stepper.h> // Essential library that performs with the stepper motor.
2 // C++ code
3
4 int bit0 = 7; // Condition that bit moves.
5 int bit_such = 1; // Bit-wise contact errors.
6 int win_hit = 12; // Bit touches winning surface.
7
8 int index = 0; // Output signal indicator for back wire contact (temporary).
9 int attack = 0; // Makes the counter increment by one and indicates back wire contact.
10 int see_count = 0; // Determines if the player is allowed to win the counter's number.
11
12 int cycle = 0; // Appears when repetition is needed.
13 int lock_event = 0; // Hold lock for end conditions.
14
15 #define STEPS 17 // Step 1: all four steps function as if they interact with a ball.
16 int step1 = 13; // Step 2: each HIGH state represent an individual coil stroke.
17 int step2 = 9; // Step 3
18 int step3 = 6; // Step 4
19 int step4 = 17; // Step 5: "STEPS" is specified in case the stepped spins very fast.
20 Stepper stepper(256, step1, step2, step3, step4); // Assigning pin variables as each coil.
21
22 int twiddle = 1;
23
24 void setup()
25 {
26   Serial.begin(9600);
27   pinMode(LED_BUILTIN, OUTPUT);
28   pinMode(bit0, INPUT);
29   pinMode(bit_such, INPUT);
30   pinMode(win_hit, INPUT);
31
32   pinMode(index, OUTPUT);
33   pinMode(attack, OUTPUT);
34   pinMode(see_count, OUTPUT);
35
36   digitalWrite(see_count, LOW);
37
38   while(digitalRead(bit_such) == LOW || digitalRead(win_hit) == LOW){
39     if(digitalRead(bit_such) == LOW || digitalRead(win_hit) == LOW){
40       while(digitalRead(bit_such) == LOW || digitalRead(win_hit) == LOW){
41         digitalWrite(LED_BUILTIN, HIGH);
42         tone(8, 330);
43         delay(400);
44         digitalWrite(LED_BUILTIN, LOW);
45         noTone(8);
46         delay(400);
47         twiddle ++; // Ungiven reserve for if you want to turn wire a certain way.
48       }
49       tone(8, 100);
50       delay(1000);
51     }
52   }
53   digitalWrite(see_count, HIGH);
54   tone(8, 750);
55   delay(100);
56   noTone(8);
57   stepper.setSpeed(256);
58 }
59
60 void loop()
61 {
62   stepper.step(twiddle);
63   delay(50); // Test to see if "motor" does "spin". Continually lower for period of function.
64
65   if (digitalRead(win_hit) == LOW) { // Contact WIN
66     Serial.println("Game Set &");
67     cycle = 0;
68     lock_event = 1;
69     while(cycle < 1){
70       digitalWrite(index, HIGH);
71       tone(8, 500);
72       delay(50);
73       digitalWrite(index, LOW);
74       tone(8, 500);
75       delay(50);
76       digitalWrite(index, LOW);
77       tone(8, 515);
78       delay(50);
79       tone(8, 754);
80       delay(50);
81       cycle++;
82     }
83     cycle = 0;
84     noTone(8);
85     while(lock_event == 1){
86       Serial.println("Win. It's working somehow.");
87       digitalWrite(LED_BUILTIN, HIGH);
88       digitalWrite(see_count, HIGH);
89       delay(1000);
90       digitalWrite(LED_BUILTIN, LOW);
91       digitalWrite(see_count, LOW);
92       delay(1000);
93     }
94   }
95 }

```

```

11 if (digitalRead(bit_couch) == LOW && digitalRead(bit0) == LOW) { // Contact Game Over
12   Serial.println("Game Set E");
13   digitalWrite(sec_count, LOW);
14   cycle = 0;
15   lock_event = 1;
16   tone(2, 131);
17   while(cycle < 4){
18     digitalWrite(indotr, HIGH);
19     delay(100);
20     digitalWrite(indotr, LOW);
21     delay(100);
22     cycle++;
23   }
24   cycle = 0;
25   tone(2, 123);
26   while(cycle < 4){
27     digitalWrite(indotr, HIGH);
28     delay(100);
29     digitalWrite(indotr, LOW);
30     delay(100);
31     cycle++;
32   }
33   cycle = 0;
34   tone(2, 134);
35   while(cycle < 4){
36     digitalWrite(indotr, HIGH);
37     delay(100);
38     digitalWrite(indotr, LOW);
39     delay(100);
40     cycle++;
41   }
42   cycle = 0;
43   noTone(2);
44   while(lock_event == 1){
45     Serial.println("Fail. It's working somehow.");
46     digitalWrite(LED_BUILTIN, HIGH);
47     delay(1000);
48     digitalWrite(LED_BUILTIN, LOW);
49     delay(1000);
50   }
51 }
52
53 if (digitalRead(bit_couch) == LOW && digitalRead(bit0) == HIGH) { // Contact Couch
54   Serial.println("ZRF");
55   tone(2, 136);
56   digitalWrite(attack, HIGH);
57   cycle = 0;
58   digitalWrite(attack, LOW);
59   while(cycle < 2){
60     digitalWrite(indotr, HIGH);
61     delay(128);
62     digitalWrite(indotr, LOW);
63     delay(128);
64     cycle++;
65   }
66   cycle = 0;
67   noTone(2);
68   Serial.println("Couch.");
69 }

```

Note that the `bit_ouch` and `win_hit` booleans are inverted due to the use of a 74HC00 (containing NAND gates).

5.2

```

#include <stepper.h> // Essential library that performs with the stepper motor.
// C++ code

int hit0 = 0; // Condition that '1' shows.
int hit_cush = 8; // Hook with contact occurs.
int win_hit = 12; // Back touches winning surface.

int indctr = 0; // Output visual indicator for hook-wise contact (temporary).
int attack = 0; // Makes the counter increment by 1 and indicates hook-wise contact.
int see_count = 11; // Determines if the player is allowed to view the counter's count.

int cycle = 0; // Appears when repetition is needed.
int lock_event = 0; // Hard lock for End Conditions.

#define STEPS 50
int steA = 10; // Step A: all four steps function as if they interact with a coil.
int steB = 9; // Step B: each HIGH state represent an individual coil invoke.
int steC = 6; // Step C
int steD = 5; // Step D: 'STEPS 10' is specified in case the stepper spins very fast.
Stepper stepper(STEPS, steA, steC, steB, steD); // Assigning pin variables as each coil.

int twiddle = 1;

void setup()
{
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(hit0, INPUT);
  pinMode(hit_cush, INPUT);
  pinMode(win_hit, INPUT);

  pinMode(indctr, OUTPUT);
  pinMode(attack, OUTPUT);
  pinMode(see_count, OUTPUT);

  digitalWrite(see_count, LOW);

  delay(100); // Intended to try and activate the while loop below.

  while(digitalRead(hit_cush) == LOW || digitalRead(win_hit) == LOW){
    if(digitalRead(hit_cush) == LOW || digitalRead(win_hit) == LOW){
      while(digitalRead(hit_cush) == LOW || digitalRead(win_hit) == LOW){
        digitalWrite(LED_BUILTIN, HIGH);
        tone(2, 350);
        delay(100);
        digitalWrite(LED_BUILTIN, LOW);
        noTone(2);
        delay(400);
        twiddle *= -1; // Ungiven remark for if you want to turn wire a certain way.
      }
      tone(2, 100);
      delay(1000);
    }
    digitalWrite(see_count, HIGH);
    tone(3, 750);
    delay(100);
    noTone(3);
    stepper.setSpeed(STEPS);
  }
}

```

```

void loop()
{
  stepper.step(itwiddle);
  //delay(1); // test to see if "motor" does "spin". Continuously lower for speed of execution.

  if (digitalRead(war_hit) == LOW) { // Contact Win
    Serial.println("Game Set A");
    cycle = 0;
    lock_event = 1;
    while(cycle < 5){
      digitalWrite(indctr, HIGH);
      tone(3, 522);
      delay(50);
      tone(3, 529);
      delay(50);
      digitalWrite(indctr, LOW);
      tone(3, 659);
      delay(50);
      tone(3, 784);
      delay(50);
      cycle++;
    }
    cycle = 0;
    noTone(3);
    while(lock_event == 1){
      Serial.println("Win. It's working somehow.");
      digitalWrite(LED_BUILTIN, HIGH);
      digitalWrite(see_count, HIGH);
      delay(1000);
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(see_count, LOW);
      delay(1000);
    }
  }

  if (digitalRead(bit_couch) == LOW && digitalRead(bit0) == LOW) { // Contact Game Over:
    Serial.println("Game Set B");
    digitalWrite(see_count, LOW);
    cycle = 0;
    lock_event = 1;
    tone(3, 131);
    while(cycle < 4){
      digitalWrite(indctr, HIGH);
      delay(100);
      digitalWrite(indctr, LOW);
      delay(100);
      cycle++;
    }
    cycle = 0;
    tone(3, 123);
    while(cycle < 4){
      digitalWrite(indctr, HIGH);
      delay(100);
      digitalWrite(indctr, LOW);
      delay(100);
      cycle++;
    }
    cycle = 0;
    tone(3, 131);
    while(cycle < 4){
      digitalWrite(indctr, HIGH);
      delay(100);
      digitalWrite(indctr, LOW);
      delay(100);
      cycle++;
    }
    cycle = 0;
    noTone(3);
    while(lock_event == 1){
      Serial.println("Fail. It's working somehow.");
      digitalWrite(LED_BUILTIN, HIGH);
      delay(1000);
      digitalWrite(LED_BUILTIN, LOW);
      delay(1000);
    }
  }

  if (digitalRead(bit_couch) == LOW && digitalRead(bit0) == HIGH) { // Contact Couch
    Serial.println("2MF");
    tone(3, 126);
    digitalWrite(attack, HIGH);
    cycle = 0;
    digitalWrite(attack, LOW);
    while(cycle < 3){
      digitalWrite(indctr, HIGH);
      delay(125);
      digitalWrite(indctr, LOW);
      delay(125);
      cycle++;
    }
    cycle = 0;
    noTone(3);
    Serial.println("Couch.");
  }
}

```

5.3

```

#include <Stepper.h> // Essential library that performs with the stepper motor.
// C++ code

int bit0 = 2; // Condition that "B" shows.
int bit_puch = 3; // Hook-wire contact occurs.
int win_hit = 4; // Hook touches winning surface.

int indctr = 5; // Output visual indicator for hook-wire contact (temporary).
int attack = 6; // Makes the counter increment by one and indicates hook-wire contact.
int see_count = 7; // Determines if the player is allowed to view the counter's number.
int prelock = 13; // Resets the list when the game is about to start.

int cycle = 0; // Appears when repetition is needed.
int lock_event = 0; // Hard lock for End Conditions

#define STEPS 101
int steA = 11; // Step A: all four steps function as if they interact with a coil.
int steB = 10; // Step B: each HIGH state represent an individual coil inwako.
int steC = 9; // Step C
int steD = 8; // Step D: "STEPS 101" is specified in case the stepper spins very fast.
Stepper stepper(STEPS, steA, steC, steB, steD); // Assigning pin variables as each coil.

int twiddle = 1;

void setup()
{
  Serial.begin(9600);
  pinMode(bit0, INPUT);
  pinMode(bit_puch, INPUT);
  pinMode(win_hit, INPUT);

  pinMode(indctr, OUTPUT);
  pinMode(attack, OUTPUT);
  pinMode(see_count, OUTPUT);
  pinMode(prelock, OUTPUT);
  digitalWrite(see_count, LOW);

  delay(100); // Intended to try and activate the while loop below.

  while(digitalRead(bit_puch) == HIGH || digitalRead(win_hit) == HIGH){
    if(digitalRead(bit_puch) == HIGH || digitalRead(win_hit) == HIGH){
      while(digitalRead(bit_puch) == HIGH || digitalRead(win_hit) == HIGH){
        digitalWrite(indctr, HIGH);
        tone(12, 880);
        delay(400);
        digitalWrite(indctr, LOW);
        noTone(12);
        delay(400);
        twiddle *= -1; // Ungiven remark for if you want to turn wire a certain way.
      }
      tone(12, 100);
      delay(1000);
    }
    digitalWrite(prelock, HIGH);
    digitalWrite(see_count, HIGH);
    tone(12, 750);
    delay(100);
    noTone(12);
    stepper.setSpeed(STEPS);
    digitalWrite(prelock, LOW);
  }
}

```

```

100 void loop()
101 {
102     stepper.step(2*middle);
103     delay(5); // Test to see if "motor" does "spin". Combining all lower for proof of function.
104
105     if (digitalRead(win_hit) == HIGH) { // Contact WIN
106         Serial.println("Game Ser A");
107         cycle = 0;
108         lock_event = 1;
109         while(cycle < 5){
110             digitalWrite(indctr, HIGH);
111             tone(12, 992);
112             delay(10);
113             tone(12, 528);
114             delay(10);
115             digitalWrite(indctr, LOW);
116             tone(12, 658);
117             delay(10);
118             tone(12, 784);
119             delay(10);
120             cycle++;
121         }
122         cycle = 0;
123         noTone(12);
124         Serial.println("Win. It's working somehow.");
125         while(lock_event == 1){
126             digitalWrite(indctr, HIGH);
127             digitalWrite(see_count, HIGH);
128             delay(1000);
129             digitalWrite(indctr, LOW);
130             digitalWrite(see_count, LOW);
131             delay(1000);
132         }
133     }
134
135     if (digitalRead(bit_cuch) == HIGH && digitalRead(bit00) == LOW) { // Contact Game Over
136         Serial.println("Game Ser B");
137         digitalWrite(see_count, LOW);
138         cycle = 0;
139         lock_event = 1;
140         tone(12, 131);
141         while(cycle < 3){
142             digitalWrite(indctr, HIGH);
143             delay(100);
144             digitalWrite(indctr, LOW);
145             delay(100);
146             cycle++;
147         }
148         cycle = 0;
149         tone(12, 129);
150         while(cycle < 4){
151             digitalWrite(indctr, HIGH);
152             delay(100);
153             digitalWrite(indctr, LOW);
154             delay(100);
155             cycle++;
156         }
157         cycle = 0;
158         tone(12, 131);
159         while(cycle < 4){
160             digitalWrite(indctr, HIGH);
161             delay(100);
162             digitalWrite(indctr, LOW);
163             delay(100);
164             cycle++;
165         }
166         cycle = 0;
167         noTone(12);
168         Serial.println("Fail. It's working somehow.");
169         while(lock_event == 1){
170             digitalWrite(indctr, HIGH);
171             delay(1000);
172             digitalWrite(indctr, LOW);
173             delay(1000);
174         }
175     }
176
177     if (digitalRead(bit_cuch) == HIGH && digitalRead(bit00) == HIGH) { // Contact Cuch
178         Serial.println("LIFE");
179         tone(12, 156);
180         digitalWrite(attack, HIGH);
181         cycle = 0;
182         digitalWrite(attack, LOW);
183         while(cycle < 2){
184             digitalWrite(indctr, HIGH);
185             delay(125);
186             digitalWrite(indctr, LOW);
187             delay(125);
188             noTone(12);
189             delay(100);
190             cycle++;
191         }
192         cycle = 0;
193         noTone(12);
194         Serial.println("Cuch.");
195     }
196 }

```

References

Basic Electronics Tutorials, n.d.. *Transmission-Gate | Pass-Transistor-Logic*. [Online]
Available at: <https://www.electronics-tutorial.net/Digital-CMOS-Design/Pass-Transistor-Logic/Transmission-Gate/>
[Accessed 18 November 2022].

Electronics Club, n.d. a. *74 Series Logic ICs*. [Online]
Available at: <https://electronicsclub.info/74series.htm>
[Accessed 2 December 2022].

Electronics Club, n.d. b. *4000 series CMOS Logic ICs*. [Online]
Available at: <https://electronicsclub.info/cmos.htm>
[Accessed 2 December 2022].

ElectronicsTutorials, c. 2019. *The Schottky Diode*. [Online]
Available at: <https://www.electronics-tutorials.ws/diode/schottky-diode.html>
[Accessed 2 December 2022].

Fairchild Semiconductor, 1999. *CD40192BC Synchronous 4-Bit Up/Down Decade Counter*. [Online]
Available at: <https://www.farnell.com/datasheets/59270.pdf>
[Accessed 14 December 2022].

LearningaboutElectronics, c. 2018. *What is an Enhancement-Type MOSFET?*. [Online]
Available at: <https://www.learningaboutelectronics.com/Articles/What-is-an-enhancement-type-MOSFET>
[Accessed 2 December 2022].

Pavankalyan, R., n.d.. *Transmission Gate And Its Truth Table*. [Online]
Available at: <https://atg.world/view-article/27093/transmission-gate-and-its-truth-table>
[Accessed 18 November 2022].

Semiconductor, H., 2003. *CD54/74HC192, CD54/74HC193, CD54/74HCT193 datasheet (Rev. F)*. [Online]
Available at: <https://www.ti.com/lit/ds/symlink/cd54hc192.pdf?ts=1671024357778>
[Accessed 14 December 2022].

Image References

Halasz, P. (2007), *File:Seven segment 02 Pengo.jpg*, Wikimedia Commons,
<https://commons.wikimedia.org/wiki/File:Seven_segment_02_Pengo.jpg> [Accessed: 18 November, 2022]

WatElectronics (2022), *7-Segment-Display-Pinout.jpg*, WatElectronics.com,
<<https://www.watelectronics.com/7-segment-display/>> [Accessed: 2 December 2022]

Semiconductor, H. (July 2003a), *CMOS BCD-to-7-Segment Latch Decoder Drivers datasheet (Rev. B)*, electronic dataset, Texas Instruments, p. 20, <<https://www.ti.com/lit/ds/symlink/cd4511b.pdf>>
[Accessed: 18 November 2022]

Semiconductor, H. (July 2003b), *CMOS BCD-to-7-Segment Latch Decoder Drivers datasheet (Rev. B)*, electronic dataset, Texas Instruments, p. 4, <<https://www.ti.com/lit/ds/symlink/cd4511b.pdf>>
[Accessed: 18 November 2022]

Semiconductor, H. (July 2003c), *CMOS BCD-to-7-Segment Latch Decoder Drivers datasheet (Rev. B)*, electronic dataset, Texas Instruments, p. 5, <<https://www.ti.com/lit/ds/symlink/cd4511b.pdf>> [Accessed: 18 November 2022]

Nave, R (n.d.), *NAND Gates*, Hyperphysics, <<http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/nand.html>> [Accessed: 2 December, 2022]

Texas Instruments (March 2022), *CD74HC93, CD74HCT93 High-Speed CMOS Logic 4-Bit Binary Ripple Counter datasheet (Rev. D)*, electronic dataset, Texas Instruments, p. 1, <<https://www.ti.com/lit/ds/symlink/cd74hc93.pdf>> [Accessed: 14 December 2022]

Video References

Haas, J. (2018), *7493 MSI Asynchronous Counters*, 31st March, YouTube. [Online Video] Available at: <https://www.youtube.com/watch?v=DziPkMy8EX0> [Accessed: 18 November 2022]

Forum References

cabraham, (September 15 2016), 'The "CD" designation indicates pin compatibility with the older 4000...' [forum], All About Circuits, <<https://forum.allaboutcircuits.com/threads/difference-between-sn-and-cd-prefixes-in-74hc-ic-series.127330/>> [Accessed: 2 December, 2022]

Fri	Mon	Tue	Wed	Thu	Fri	Mon	Tue	Wed	Thu	Fri	Mon	Tue	Wed	Thu	Fri	Mon	Tue										
4	7	8	9	10	11	14	15	16	17	18	21	22	23	24	25	28	29	30	1	2	5	6	7	8	9	12	13

November

2022

2702

[illegible]

[illegible]

April

2023

May

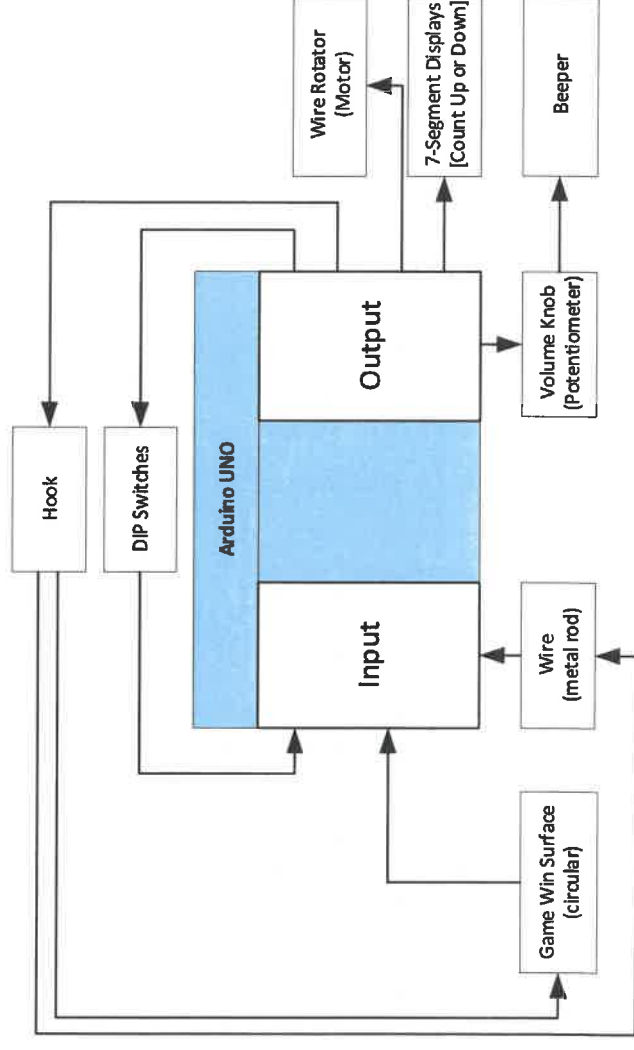
2023

[illegible]

Mon	Tue	Wed	Thu	Fri																												
26	27	28	29	30																												

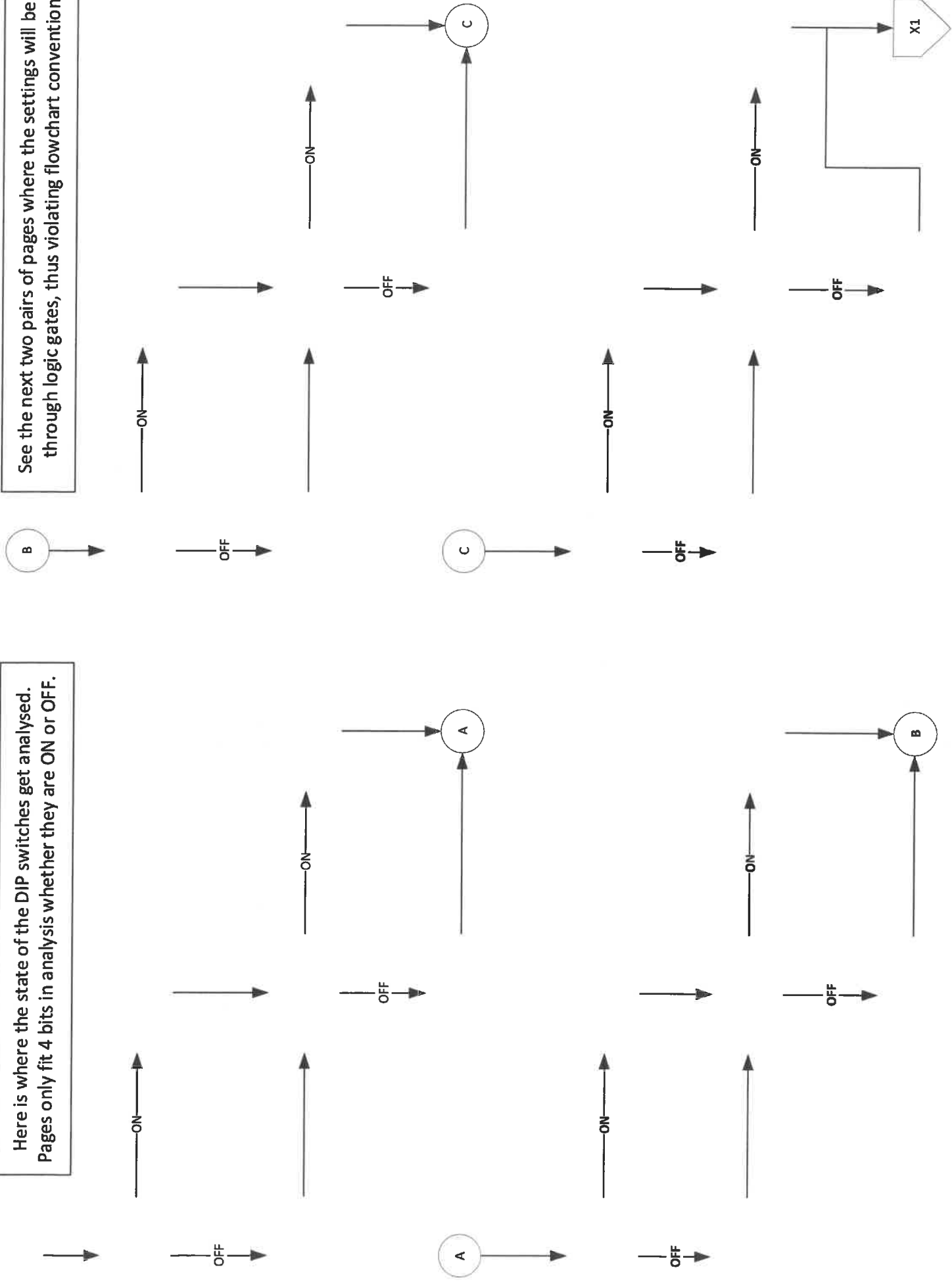
Wire Game by Ronan Pesarillo

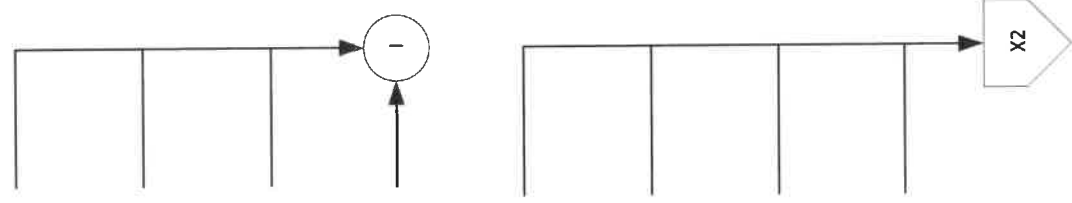
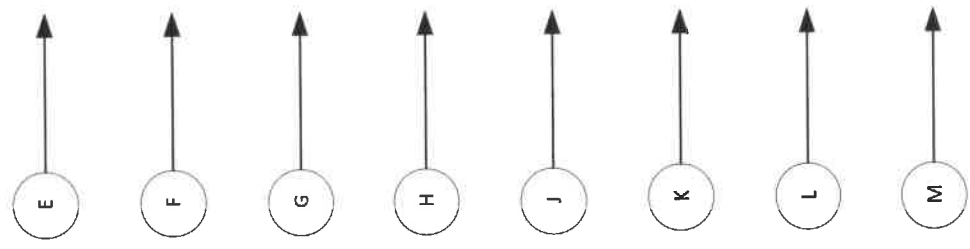
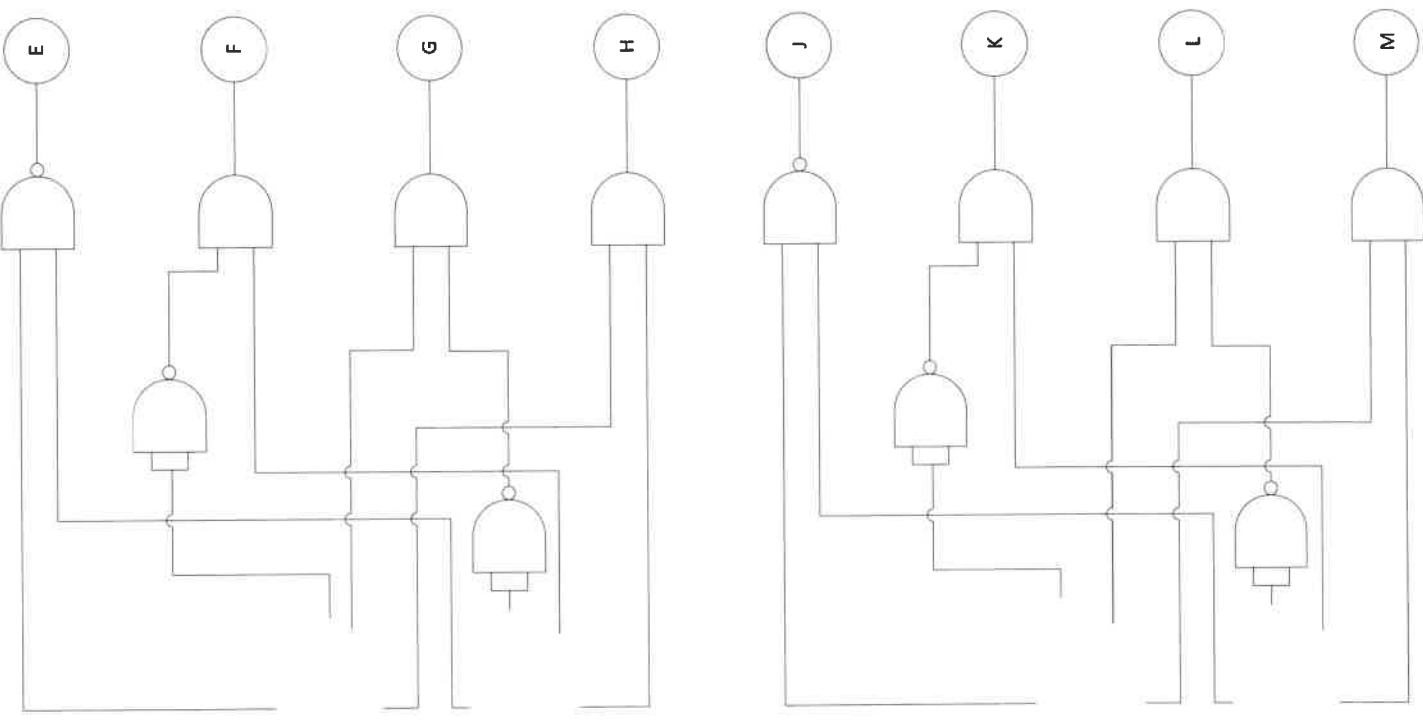
Block Diagram

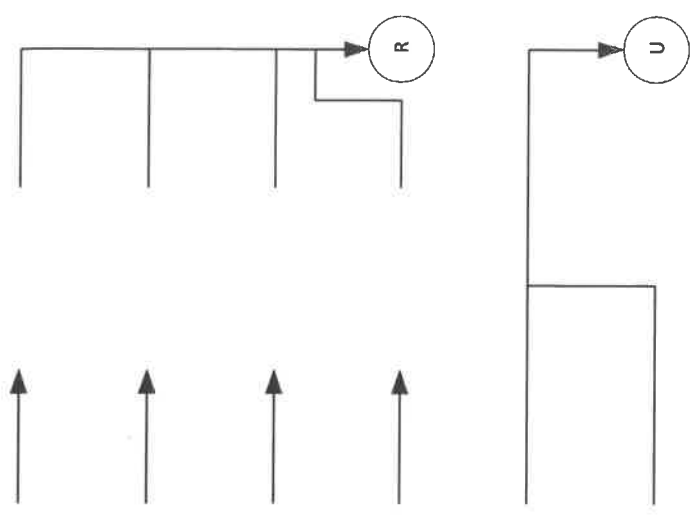
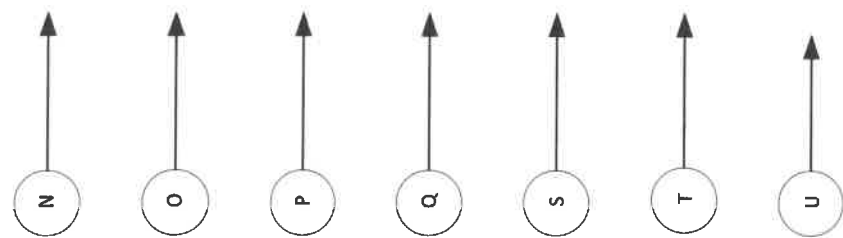
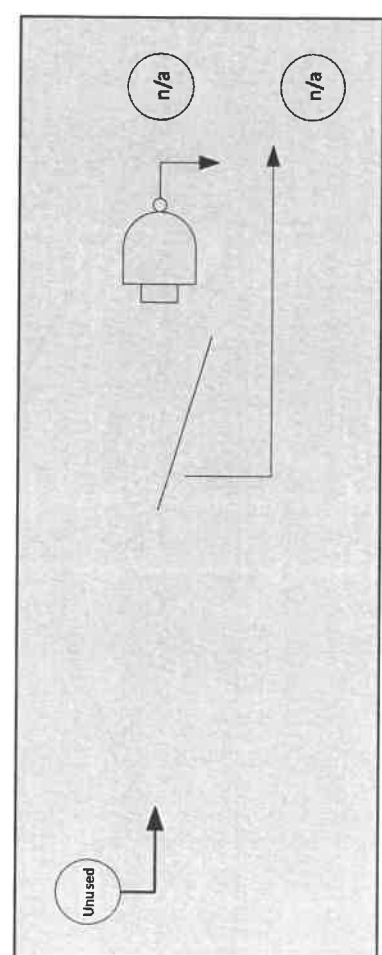
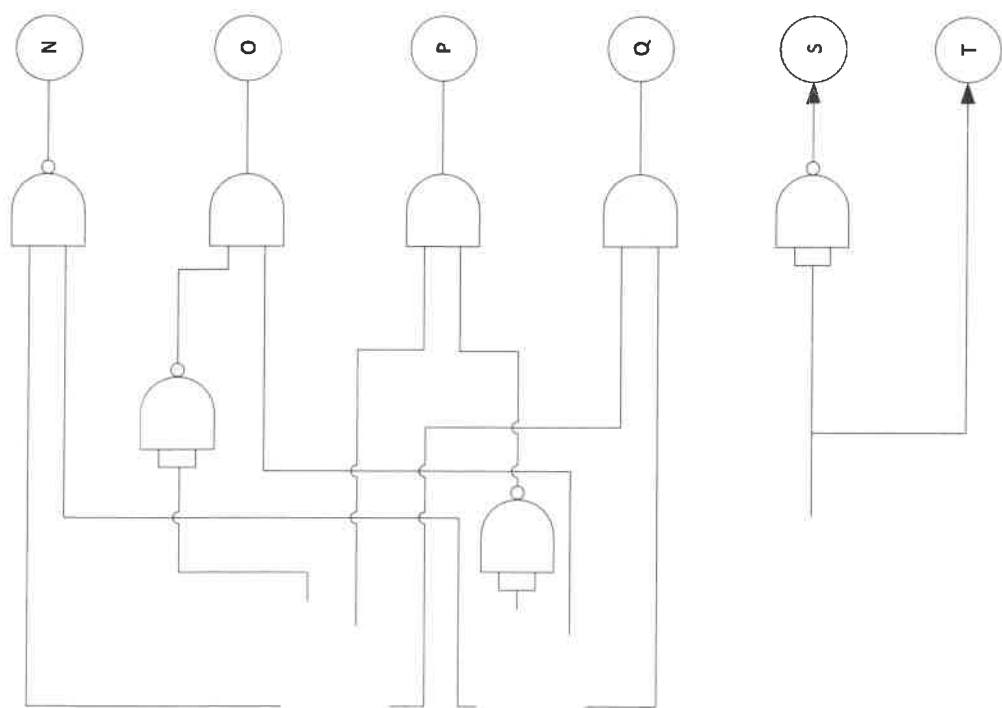
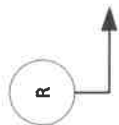


Here is where the state of the DIP switches get analysed.
Pages only fit 4 bits in analysis whether they are ON or OFF.

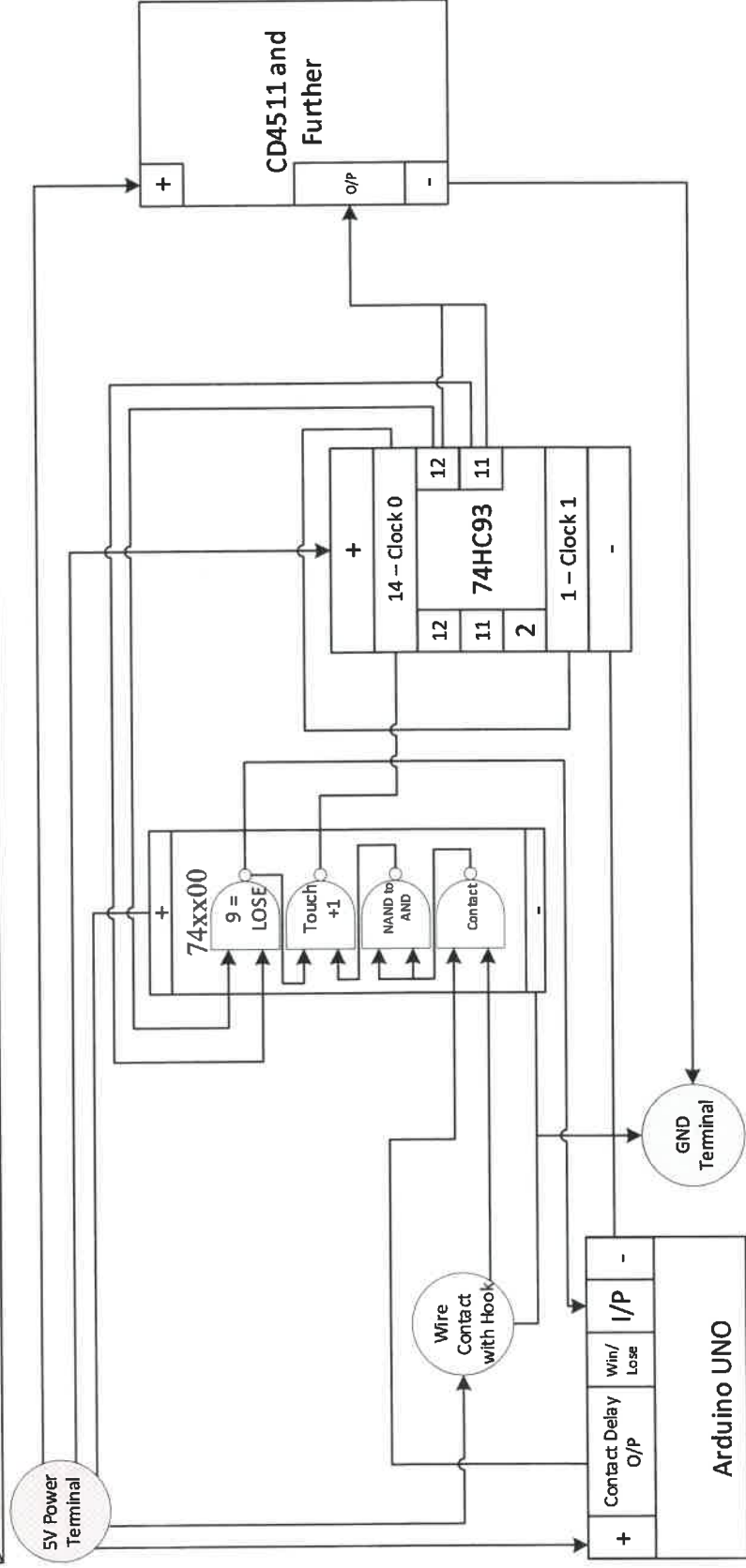
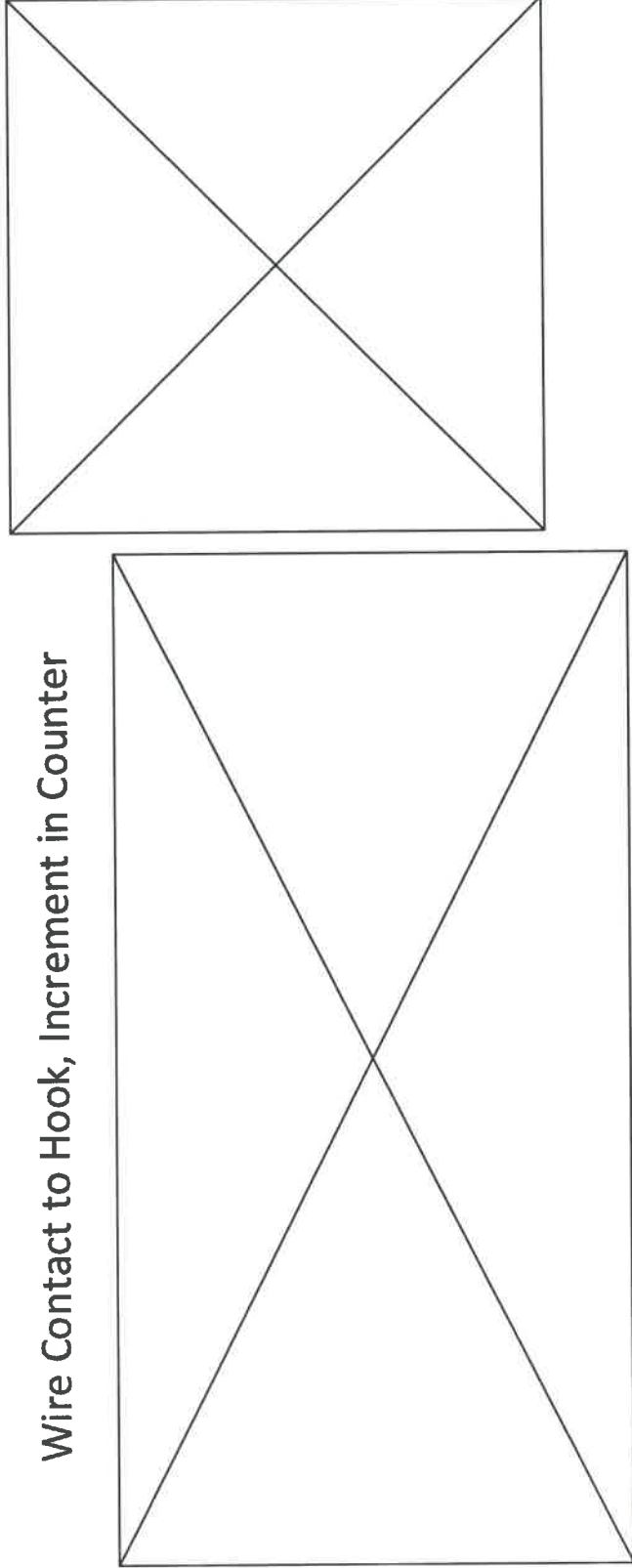
See the next two pairs of pages where the settings will be set
through logic gates, thus violating flowchart conventions.



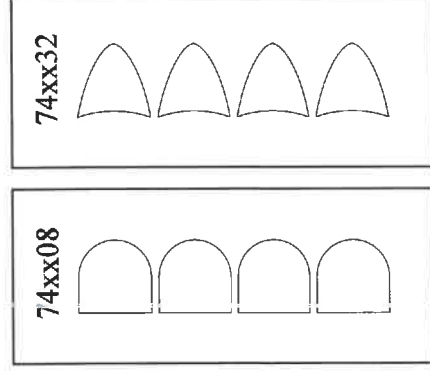
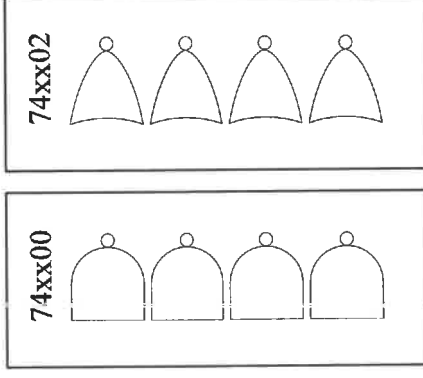




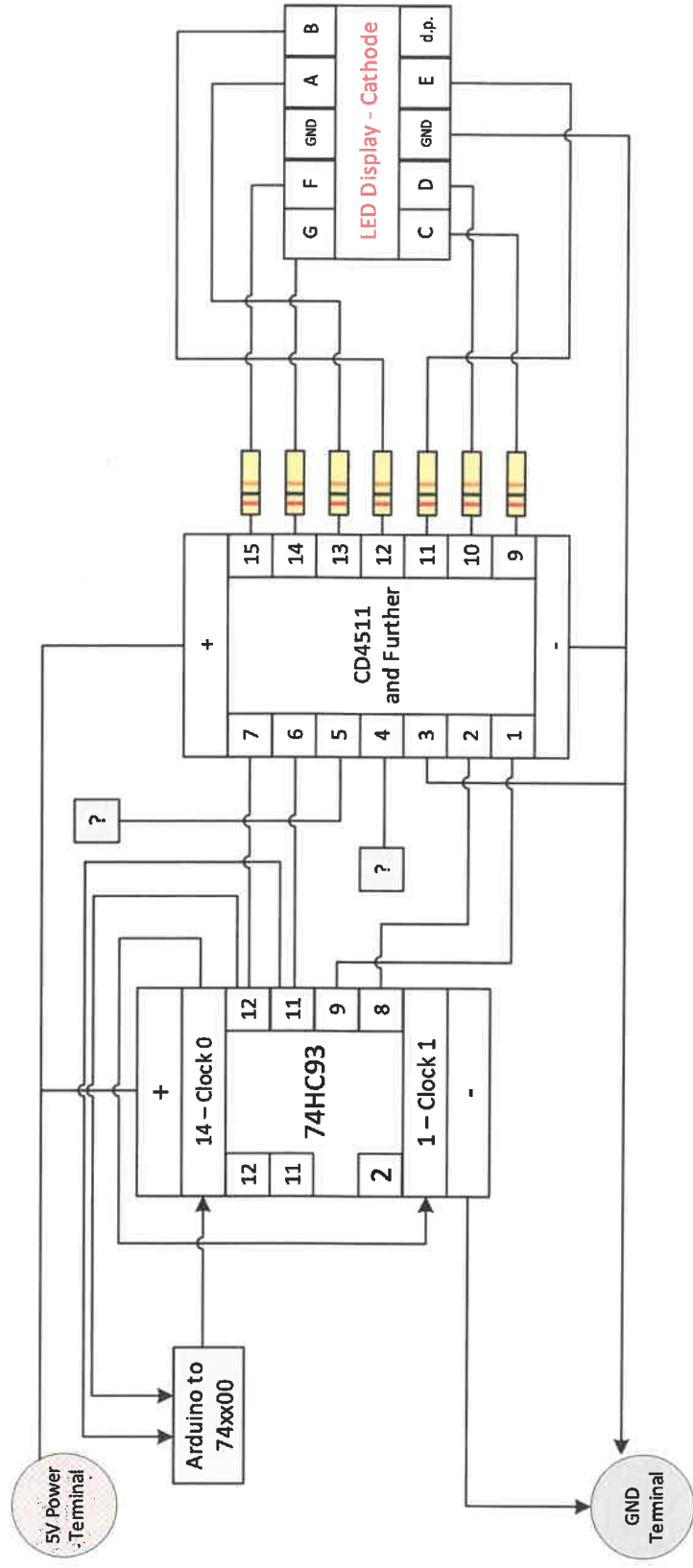
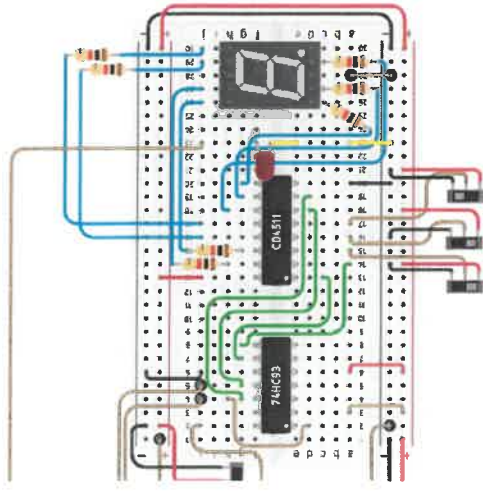
Wire Contact to Hook, Increment in Counter

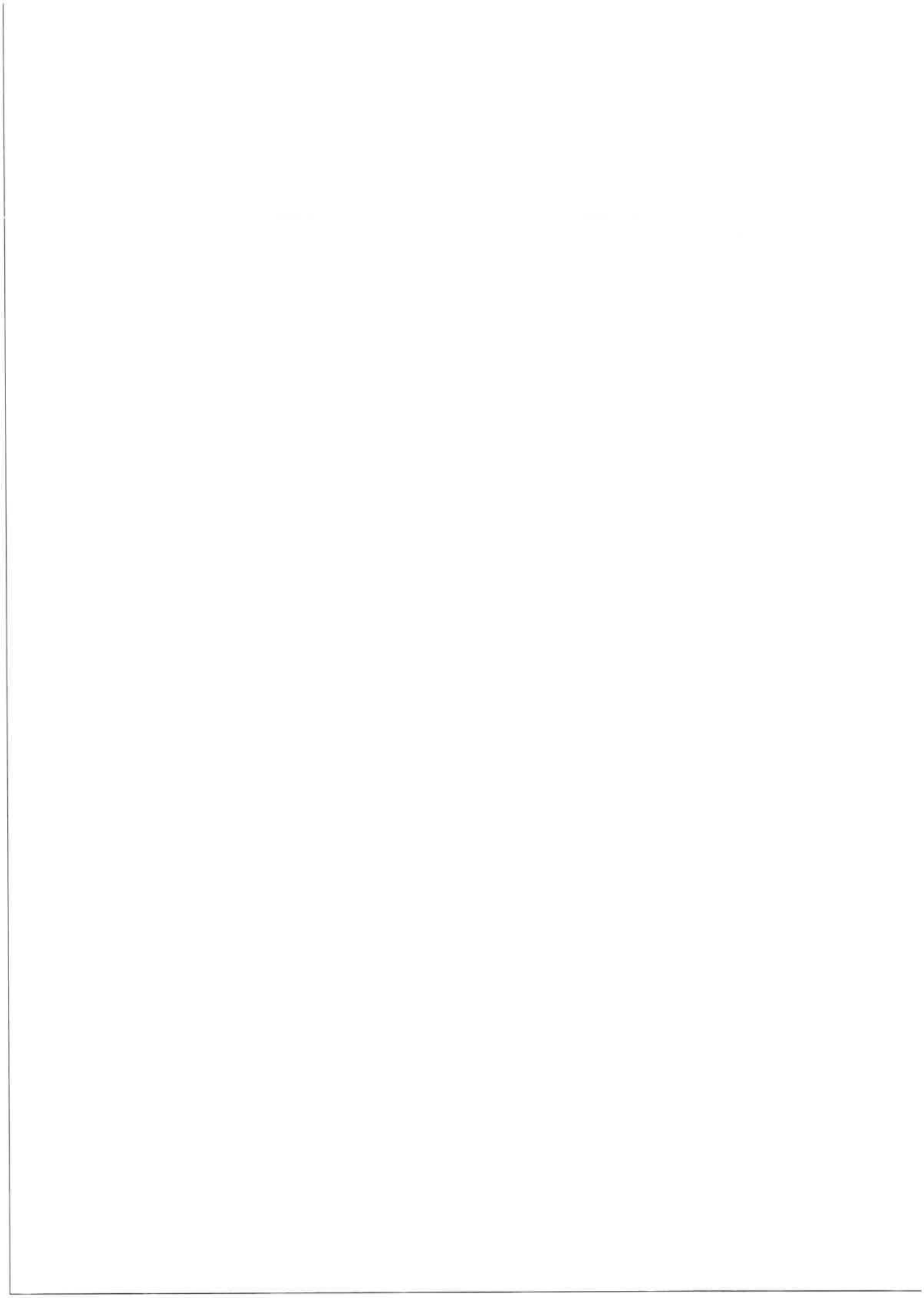


74xx00 – Quad 2-NAND
 74xx02 – Quad 2-NOR
 74xx08 – Quad 2-AND
 74xx32 – Quad 2-OR



Binary to Decimal Counting, One Digit





Append - List of Projects as Initial Decisions

Contents

1. Paper Piano with Arduino by Barqunics – © GPL3+ From create.arduino.cc.....	2
2. How To Make Alcohol Breathalyser Circuit? From Electronics Hub.....	3
3. Fidget Spinner RPM Counter by Andriy Baranov – © GPL3+ From create.arduino.cc.....	4
4. Buzz Wire with Score Counter by behrooz66 – © MIT From create.arduino.cc	4
References.....	5

1. Paper Piano with Arduino by Barquonics – © GPL3+ From create.arduino.cc

Connected/clipped surfaces like pencil's graphite area can be connected to play a tune like a piano or a simple synth.

- The Paper Piano plays tunes by touching respective graphite fields drawn on a piece of paper with a human limb or joint to make a sound emitted by the Arduino from the speaker.
- This project utilises an existing library that detects touch or proximity – CapacitiveSensor. Its function may be similar to an invention kit named Makey Makey.
- Resistor values for the recipient pin listed are guidelines that up to a 1 (or less) megaohm resistor will sense human touch unless higher resistor values increase touch proximity of the graphite fields. And coded thresholds for each sensor prevent unnecessarily extreme sensitivity to play tones prematurely.

[link outdated, redirects to <https://projecthub.arduino.cc/>]

2. How To Make Alcohol Breathalyser Circuit? From Electronics Hub

Measure how much alcohol is in the breath tester's blood.

- Breathalysers are used by police officers throughout the world to enforce drink driving laws. In the 1930s is when the USA used them by their law enforcement and were introduced in 1967 in the UK.
- They can be used to carry out preliminary and evidential breath tests.
 - A reading below 35 or 40 micrograms per 100ml of a breath of alcohol will release a suspect from drunk driving. Between 40 to 50 μ g/100ml acquires the suspect to take a blood test. Along with significant difference in reading, the suspect will be given the choice for their urine to be sampled instead. And if the reading is over 50 μ g/100ml, then the suspect will be charged correspondingly.
- Breathalysers come in different builds such as: colour-changing solutions; components that include wires; and electronic components that determine chemical density.
 - The earliest build, though its shape spans approx. a shoe box named 'The Drunkometer' developed by Professor Rolla N. Harger in Indiana, USA where a motorist's breath is sampled into a balloon and is pumped through an acidified potassium permanganate solution. The greater the change in colour, the more alcohol there was present in the breath. (Drink Driving .org, n.d.)
- Modern store-bought breathalysers can be used as a portable equipment for holidays where alcohol-engaged events or locations can leave the tourist into trouble if they cannot regulate themselves.
- The main component for the circuit, the MQ-3 alcohol gas sensor contains a material called 'tin(IV)/stannic oxide' which is suitable for detecting chemicals such as: alcohol, benzene, CH₄, Hexane, LPG (low-petroleum gas), and CO. This specific gas sensor is mostly used to measure alcohol, ethanol and smoke. (Components 101, 2018)
- The MQ-3 alcohol gas sensor contains a "disadvantage" that I could take advantage of, which is the sensitivity of other chemicals not meant to be part of a tester's breath.
 - If anything, I could set its sensitivity high enough so to measure the strength of air irregularities. This was meant to be the job of a component called the 'MQ-135 Gas sensor' and it may also be purchased alongside for component comparison tests.
- The project will also contain an LM3914 that is often used to visualise the intensity of a signal. Its 9th pin, 'MODE', chooses whether or not it will 'switch ON' only one pin – 'dot mode' if OFF, or it will 'switch ON' a number of signals at once – 'bar mode' if ON.

(ElectronicsHub, 2016)

3. Fidget Spinner RPM Counter

by Andriy Baranov – © GPL3+

From create.arduino.cc

This project measures the rotations per minute of a spinning object. In this project, a fidget spinner is used.

- The Fidget Spinner RPM Counter measures how many “revolutions per minute” does the Fidget Spinner rotate.
- The project page contains a video demonstrating the project, and two lists of components, yet the second list contains a neodymium magnet that has to be shaped small enough to attach to one spoke of the fidget spinner.
- This project introduces the magnetic concept called the ‘Hall Effect’ where it carries an electric current and is placed in a magnetic field perpendicular to the current causing development of a transverse electric field in a solid material. (Britannica, T. Editors of Encyclopaedia, 2021)

[link outdated, redirects to <https://projecthub.arduino.cc/>]

4. Buzz Wire with Score Counter

by behrooz66 – © MIT

From create.arduino.cc

An interactive metal-to-metal game in which you guide a wand across a metal outline.

- There appears to be no clear history as to who invented this game.
 - The game itself is simple to build with the fundamental tools and materials, and the metal outline can come in a variety of shapes and sizes (depending on the manufacturer), leaving countless possibilities for obstacle design and even the number of mechanics to implement.
 - I could implement millisecond-based timing for a more precise time reading once the player is done guiding their loop around the line.
 - I may add an extra 7-segment display for at least ten times more countable contacts/touches/mistakes registered to count in the game. Plus, with the changing of code to ensure the Arduino counts.
 - An extra piece of metal that the wire touches as an alternate time-stopping output and sound a different tone/tune from the buzzer.
 - I am planning to create a Hamiltonian-pattern cube of 4 cubic meters or a Hamiltonian tesseract. That is if this project choice gets accepted.
 - One of the simplest possible builds can just use a battery, a sturdy metal wire, a flexible piece of wire, a resistor, and an outputting component (e.g: an LED).

[link outdated, redirects to <https://projecthub.arduino.cc/>]

References

Britannica, T. Editors of Encyclopaedia, 2021. *Hall effect*. [Online]

Available at: <https://www.britannica.com/science/Hall-effect>

[Accessed 19 October 2022].

Components 101, 2018. *MQ-3 Alcohol Gas Sensor*. [Online]

Available at: <https://components101.com/sensors/mq-3-alcohol-gas-sensor>

[Accessed 19 October 2022].

Drink Driving .org, n.d. *ALCOHOL BREATHALYZERS*. [Online]

Available at: https://www.drinkdriving.org/drink_driving_information_breathalysers.php

[Accessed 19 October 2022].

ElectronicsHub, 2016. *How To Make Alcohol Breathalyzer Circuit ?*. [Online]

Available at: <https://www.electronicshub.org/breath-alcohol-tester/>

[Accessed 25 October 2022].

