国家資格第二種電気工事士資格取得支援のためのスマートフォンアプリ開発

Development of a Smartphone App to Support the Acquisition of a National Qualification as a Class 2 Certified Electrician

貞方 敦雄[1]

概要：

理工学部電気工学科では、2018 年度より電気工事実習という実習科目で国家資格である第一種及び第二種電気工事士の資格取得支援を行っている。電気工事士の試験内容は、筆記試験と技能試験に別れている。筆記試験対策では、模擬試験問題を出題、採点後、学生の理解度を試験問題結果より分析し、苦手とする問題について解説を行っている。しかし、履修生が 40 名程度になると集計と分析が間に合わず適切な指導が困難である。本研究開発の目標は、第二種電気工事士資格取得支援のためのオリジナルスマートフォンアプリを開発し、上記の課題などを解決し、資格取得率を向上させることである。本投稿は第二種電気工事士の筆記試験対策用のスマートフォンアプリの開発に向けた取り組みを紹介する。

Abstract :

Since 2018, the Department of Electrical Engineering at the Faculty of Science and Technology has been supporting the acquisition of national qualifications as Class 1 and Class 2 electricians in a practical course called Electrical Engineering Practice. The electrician examination consists of a written examination and a skills examination. In preparation for the written examinations, mock examinations are given and scored, the students' level of understanding is analyzed based on the results of the examinations, and explanations are given on problems they have difficulty with. However, when the number of students in the course reaches around 40, it is difficult to collect and analyze data in time, making it difficult to provide appropriate instruction. The goal of this research and development is to develop an original smartphone app to support the acquisition of electrician qualifications, solve the above issues, and improve the qualification acquisition rate. This paper introduces our efforts toward the development of a smartphone app to prepare for the written test for second-class electricians.

キーワード：電気工事士，資格取得支援，アプリ開発

Keywords：Electrician, Qualification acquisition support, App development

[1] 理工学部 電気工学科 講師

1 はじめに

国家資格である電気工事士の資格取得を掲げた「電気工事実習」という 4 年生大学では非常に珍しい実習科目の設置経緯について紹介する。理工学部電気工学科では、2018 年度より電気工事実習という実習科目で国家資格である第一種及び第二種電気工事士の資格取得支援を行っている。電気工事士の資格取得支援に特化した授業として立ち上げる以前は、当時工学部電気工学科の退職された大坪助手らが中心となり、電気工学科の学生の有志が数名集まって課外の時間で主に技能試験対策が行われていた。この時代は、学生が自主的に集まり、第一種や第二種電気工事士の技能試験合格に向けて取り組まれていた。そのため、電線や器具等を購入する予算はほぼ無く、電気工学実験などの余った予算や学生の手出しによって実施されていた。また、専用の実習場所は無く、学生実験室の片隅を借り、先生が学生に指導、学生が空き時間に練習のために利用していた。この様に、指導環境としては好ましいとは言えない中、毎年、数名の電気工事士の資格取得者が誕生していたことに驚きである。当時、資格を取得した学生は、学科の成績上位者でも入社することが難しかった九電工をはじめとする大手の電気工事会社に就職していた。この様な背景があり、学生のキャリア支援の一環として、工学部から理工学部に再編されるタイミングで、正課の授業として第一種及び第二種電気工事士の資格取得支援を専門に教える実習科目として貞方が担当している「電気工事実習」を設立した。現在では、電気工事士の技能試験対策を行う専用の実習室や予算、指導員として同学科の梅崎技能員、森技能員、山光助手、有資格者で構成される学生アルバイタの学生により、充実した実習体制を構築することができている。近年では、電気工事実習の正課及び授業外の正課外での指導を合わせると年間で 20 名を超える学生が第一種や第二種電気工事士に合格することができるようになった。この場を借りて、尽力頂いている関係者の方に感謝申し上げます。

本報告の構成について説明する。第 2 章で、電気工事士の概要について紹介する。特に、本報告は総合情報基盤センターの報告書として出版されるため、多くの読者は電気工事について馴染みが無いと思われるため、少し分量を多く執筆している。第 3 章では、理工学部電気工学科の電気工事実習での取り組み内容について説明する。この章では、本投稿の中心である筆記試験対策での課題点などを挙げる。そして、スマートフォンアプリとして、指導する立場から見て効率よく受講生の理解度を把握し、解説等を通じて学生の理解向上を促すことができるか提案する。第 4 章では、今回のスマートフォンアプリ開発で用いたプラットフォームである Flutter について紹介する。第 5 章では、Flutter を用いたアプリ開発の例として、クラウドシステムを利用したメモアプリについて説明する。第 6 章では、電気工事士筆記試験対策アプリについての概要や開発途中の内容についてコードや画面を示しながら説明を行う。最後に、第 7 章では、本研究課題のまとめと今後の課題を示している。

2 電気工事士の概要

本章では、国家資格である第一種及び第二種電気工事士の資格試験の概要や必要性について、特に、新しい都市計画として進んでいるスマートシティー・スマートハウス、電気工事士の人材不足等に触れながら説明を行う。その後、電気工事士の資格取得支援を行っている電気工事実習での取り組み内容を紹介する。

2.1 電気工事士の資格

電気工事の欠陥による災害を防止するために、電気工事士法によって一定範囲の電気工作物について電気工事の作業に従事する者の資格が定められている[1]。電気工作物とは、電気を供給するための発電所、送電する電圧を変える変電所、送配電線路をはじめ、工場、ビル、大学などの施設、住宅等の受電設備、屋内配線、電気使用設備などの総称である。電気工作物は、一般用電気工作物等、事業用電気工作物、自家用電気工作物に別れている。

①一般電気工作物等は、主に一般住宅や小規模な店舗、事業所などのように、他の者から低圧 (600 ボルト以下) の電圧で受電している場所等の電気工作物を指す。

②事業用電気工作物は、一般用電気工作物等以外の電気工作物を指す。

③自家用電気工作物は、事業用電気工作物であって、電気事業 (一定規模以下の発電事業を除く) の用に供する電気工作物以外の例えば工場やビルなどのように電気事業者から高圧以上の電圧で受電している事業所等の電気工作物 (需要設備等) を指す。

電気工事士の資格には、第一種電気工事士と第二種電気工事士があり、第一種電気工事士では、一般用電気工作物等及び自家用電気工作物 (最大電力 500 キロワット未満の需要設備に限る) の作業に、第二種電気工事士では、一般用電気工作物等の作業にそれぞれ従事することができる。なお、第一種及び第二種電気工事士の資格試験は、一般財団法人　電気技術者試験センターが試験実施機関として行っている[1]。同センターでは、電気工事士以外にも電気技術者として重要な国家資格である電気主任技術者についても実施している。

2.2 電気工事士資格の必要性

何も考えずに家電製品や電気を使った豊かで便利な暮らしを営めるのは、ひとつに国家資格として第一種及び第二種電気工事士の資格制度が制定されているためである。東京消防庁の調査によると、令和 3 年中の東京消防庁管内では、3,935 件の火災が発生し、そのうち電気設備機器などによる火災 (以下、電気火災という) は、1,399 件 (前年比 236 件増加) と、全火災件数の 35.6 % を占めている[2]。電気火災の主な原因としては、電気や電気製品を使用する際の「不適切な維持管理」(例えば、トラッキング現象による出火。トラッキング現象とは、コンセントに差し込んだプラグの差し刃間に付着した綿埃等が湿気を帯びて微小なスパークを繰り返し、やがて差し刃間に電気回路が形成され出火する現象) や「取り扱い上の不注意」

(例えば、電源タップに定格電力以上のコンセントを接続して使用する「たこ足配線」による出火など) である。これらは電気火災の原因の約 6 割を占めている。「不適切な維持管理」や「取り扱い上の不注意」に関しては、利用者が正しい知識を身に付けて使用することで防ぐことができる。一方で、利用者の落ち度ではない原因で発生した電気火災の内訳を見ると、「設置（取付）工事方法不良」が 66 件、「構造機構不良・改悪する」が 65 件と合計で 131 件もの電気火災が発生している。東京消防庁管内では、約 1 割が電気工事絡みの電気火災が起きていることになる。したがって、財産や人の命を奪いかねない電気工事については、従事者に対して正しい知識や技能、責任を持って施工して頂けるように国家資格として資格を持った者でなければ電気工事を行えないように法整備等が進められてきた。

電気機器の利用者だけでなく電気工事の従事者に多い事故として感電があげられる。労働安全衛生総合研究所が感電に関しての基礎と過去 30 年間の死亡災害の統計をまとめている [3]。感電の何が一番危険か説明する。例えば、洗濯機を利用している時に、洗濯機が故障して電流が流れてはいけないところに漏電していたとする。この状況で、利用者が洗濯機に触れると感電が発生する。感電が与える人体への影響は、心臓にどのくらいの電流が流れるのか、その通電時間はどのくらいなのかで生死を分けることになる。感電の危険性として、4 段階の DC-1 から DC-4 に別れている。DC-3 では、数百 mA の電流が数秒間人体に流れることで、心臓に回復可能な障害と伝達障害が起きる可能性があるとされている。DC-4 では、DC-3 より大きな電流が心臓を流れることで、危険な病理生理学上の症状、例えば重度の火傷や心室細動が起きる確率が約 50 % 以上になるなど、死亡するケースも出てくる。電気工事では、漏電による災害が起きないように、漏電遮断器の利用や接地工事などの手法を用いて防止している。そのため、洗濯機などの利用にあたっては、緑色のアース線をコンセントに付いているアース端子にしっかりと接続していれば、感電を防ぐことができる。そのため、利用者の感電による死亡件数は少なくなっている。

一方で、電気工事に従事する者が作業中に誤って感電してしまい死亡する件数が昔は多かった。1960 年代では、年間に約 400 名もの作業者が感電によって死亡していた。そのため、1960 年に電気工事法の制定に伴い、第二種電気工事士　一般用電気工作物工事従事者の免許義務化が始まった。1962 年には高圧電気工事技術者試験 (任意制) が発足、1964 年には電気事業法が制定されている。法整備等により感電による死亡者数は減少し、1980 年代には年間 100 名を下回る程度までになった。しかし、依然として多い状況が続いていた。そのため、1987 年に第一種電気工事士として自家用電気工作物工事従事者の免許義務化が開始された。2000 年代に入っても感電による死亡者数はゼロではく、年間 10 名程度となっている。したがって、各企業で実施されている安全教育などが一層重要である。大学での電気工事士に関する教育では、感電をはじめとした電気に関する基礎知識をしっかりと教え、何が危ないのか、どの様にしたら回避できるのか理論的に考えることができるように教育活動に取り組む

必要がある。この点については、電気工学を体系的に学ぶ4年生大学で電気工事士の資格取得支援をする意義でもあると考えている。なぜなら、大学を卒業後に電気工事関連の仕事に就く際は、基本的には、現場を監督する立場である施工管理者として勤務する。施工管理者は、電気工事従事者が安全に作業を行えるか監督する責任が生じる。ゆえに、電気工学に関する正しい知識等を有して、労災を未然に防ぐには4年制大学で学ぶ電気工学に関する学問の知識が活かされると考えているからである。

近年、第一種及び第二種電気工事士の人材不足が電気工事業界の課題として挙げられている[4]。電気工事に従事されている高齢者層の退職により、2045年には業界全体として第一種電気工事士が2.4万人、若い世代の入職者の減少等により第二種電気工事士が0.3万人不足する予想が出されている。そのため、電気工事業界からは、高校、高専、大学などの教育機関に対して、電気工事業界への関心を持たせ就職先として選択して頂けるような教育活動やキャリア支援が望まれている。

電気工事士が活躍する場は、益々広がっている。その一例として、国土交通省などが主導している「スマートシティ官民連携プラットフォーム」を紹介する[5]。サイバーとフィジカルを高度に融合したSociety 5.0の実現に向け、AI、IoTなどの新技術やデータを活用したスマートシティをまちづくりの基本コンセプトとして位置付け、スマートシティの取組を官民連携で加速するためプラットフォームである。課題先進国である日本では、急速な高齢化、多発する都市型災害など世界各国の多くの都市がいずれ直面する都市課題に先んじて直面しており、これらを解決する一つの政策としてスマートシティやスマートハウスの実現と普及に向けて産学官連携で取り組まれている。今後、全国的にスマートシティやスマートハウスが普及することを考えると、電力インフラ設備、通信インフラ設備、都市開発、改修･補修工事、EV車専用充電スタンドなどの新たな電気工事の需要が生み出され、電気工事士が活躍する場が増加する一方で、人材不足をどの様に解消していくかが課題となる。

上記の背景から、理工学部電気工学科では、電気工事士として社会貢献できる技術者を育成するために「電気工事実習」という実習科目を理工学部新設時に設け、2018年度より正課の授業として学生へ指導を行っている。

2.3 第二種電気工事士の試験内容

ここでは、第二種電気工事士の試験内容についてのみ説明する[6]。第一種電気工事士の試験内容に関しては参考文献をご覧いただきたい[7]。第二種電気工事士の試験は、一般電気工作物等の保安に関して必要な知識及び技能について、筆記試験及び技能試験で評価される。なお、正式には筆記試験ではなく学科試験であるが、多くの電気工事士に関する書籍等では筆記試験と書かれている[8]。そのため、本投稿では、筆記試験と記することにしている。第二種電気工事士の試験日程は、上期と下期試験と年に2回実施される。筆記試験に合格した

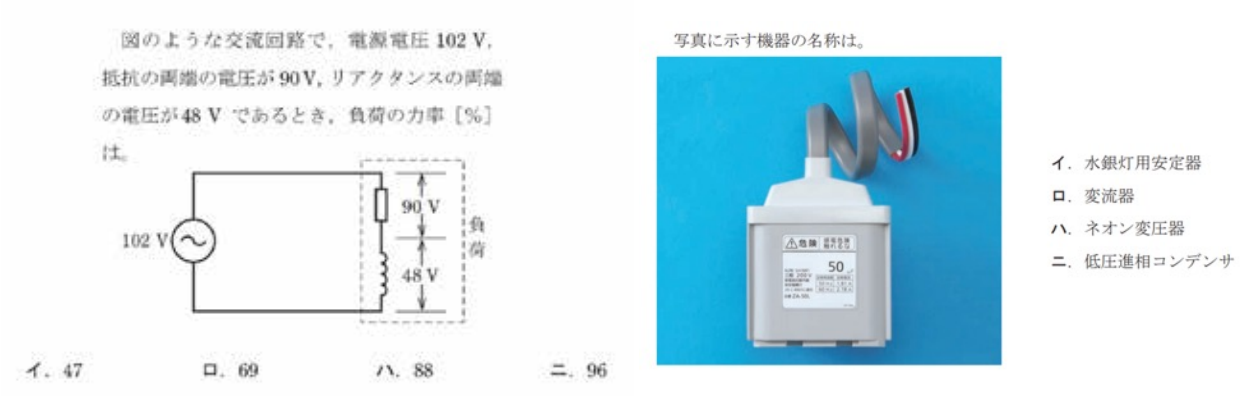
者が、技能試験を受験することができ、技能試験まで合格した者が、国家資格である第二種電気工事士の免状を頂くことができる。なお、筆記試験申込者及び筆記試験免除対象者は、上期試験、下期試験の両方の受験が可能である。筆記試験免除対象者は、前回の第二種電気工事士筆記験に合格した方や高等学校、高等専門学校及び大学等において経済産業省令で定める電気工学の課程を修めて卒業した方などが対象である[9]。九州産業大学理工学部電気工学科は所定の科目の単位を修得し卒業することで筆記試験免除となる。ただし、在学中は、筆記試験免除とならないため、通常は、筆記試験から受験することになる。技能試験の免除制度はない。

(1) 筆記試験

次に掲げる内容について関する内容が合計 50 問出題され、解答方式はマークシートに記入（筆記方式）又はパソコンで解答（CBT 方式）する四肢択一方式で実施されている。

(ア) 電気に関する基礎理論
(イ) 配電理論及び配線設計
(ウ) 電気機器・配線器具並びに電気工事用の材料及び工具
(エ) 電気工事の施工方法
(オ) 一般用電気工作物等の検査方法
(カ) 配線図
(キ) 一般用電気工作物等の保安に関する法令

幾つか実際に出題された問題を取り上げて、どの様な形式の問題が出題されているのか説明する[10]。図 1(1) に示す様な、電気回路という電気工学科の基礎となる科目で習う内容に関する計算問題が出題される。また、図 1(2) に示す様な電気機器の写真を見て名称等を選択する写真鑑別問題が出題される。特に、基礎理論である計算問題は、毎回 8 問程度出題されている。また、電気機器や工具などの写真鑑別問題も多く出題される。この問題については、比較的簡単に回答できるが写真と名称や用途などを覚えておく必要がある。そのため、上記の内容を勉強することができる機能を有したスマートフォンアプリを作成することとした。



(1) 基礎理論問題　　(2) 写真鑑別問題

図 1 基礎理論及び電気機器に関する問題例

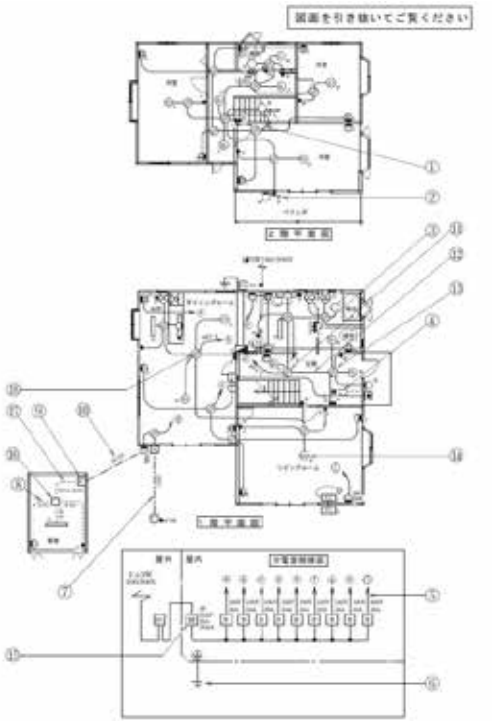


図 2 配線図問題

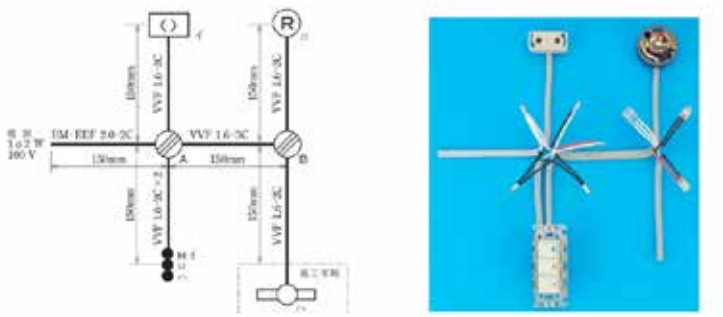
筆記試験の 50 問中 20 問は配線図に関する問題が出題される。配線図問題では、実際の電気工事を想定し、図 2 に示す様な、住宅や小規模事務所、倉庫、工場、コンビニなどの建築図面・間取り図に電気配線図が描かれた図面を見ながら回答する形式である。通常試験では A3 サイズの問題用紙に図面が描かれている。今回はスマートフォンアプリを用いて筆記試験対策を行うため、小さなスマートフォンの画面でも配線図を見やすくする工夫が必要となった。詳細については、後半の章で説明する。

(2) 技能試験

筆記試験の合格者と筆記験免除者に対して、次に掲げる事項の全部又は一部について実施される。試験は、持参した作業用工具により、配線図で与えられた問題を支給される材料で、一定時間内に完成させる方法で行う。第二種電気工事士の技能試験は試験時間 40 分である。

(ア)　電線の接続
(イ)　配線工事
(ウ)　電気機器及び配線器具の設置
(エ)　電気機器・配線器具並びに電気工事用の材料及び工具の使用方法
(オ)　コード及びキャブタイヤケーブルの取付け
(カ)　接地工事
(キ)　電流、電圧、電力及び電気抵抗の測定
(ク)　一般用電気工作物等の検査
(ケ)　一般用電気工作物等の故障箇所の修理

技能試験問題は、図 3 に示す様な配線図及び施工条件等を読み、単線図から複線図へ書き直し、実際に電線を切断、被覆を剥き、電線同士を接続、器具への電線の取付等の作業を行う。そして、試験時間内に欠陥なく完成させる必要がある。技能試験の問題は、毎年公表問題として 13 課題が周知されており、各試験会場で、どの公表問題が出題されるかはランダムになっている [11]。



(1) 配線図　　(2) 完成状態

図 3 技能試験問題例

3 電気工事実習の取り組み紹介及び試験対策の課題点

本章では、理工学部電気工学科の 2 年次後期開講の実習科目である電気工事実習での第一種及び第二種電気工事士の資格取得支援の取組について紹介する。また、本研究開発にて電気工事士の資格取得支援を行うアプリ開発を始めるきっかけとなった試験対策での課題点等についても説明する。

3.1 試験対策のスケジュール (授業日程)

電気工事実習のシラバスに記載している講義概要は、『国家資格である第一種、第二種電気工事士の取得に強い関心が有り、また、将来電気工事関連の企業で働きたいと考えている学生を対象に、筆記試験や技能試験対策として基礎から応用までを演習及び実習を通じて学ぶ。本実習を真剣に取り組むことで第一種、第二種電気工事士を合格するために必要な知識や技能が身に付く。』としている。そのため、電気工事士の試験日程 (第一種 : 筆記試験 10 月、技能試験 12 月、第二種 : 上期 (筆記試験 5 月、技能試験 7 月)、下期 (筆記試験 10 月、技能試験 12 月)) に合わせた授業日程となるように授業日が変則的となっている。図 4 に電気工事士資格取得支援スケジュールを示す。正課の授業としての資格取得支援は後期開講の電気工事実習で履修者に指導を行う。第一種電気工事士の資格取得を希望する学生も同時に指導している。第一種電気工事士の資格取得を目指す学生は前年度に第二種電気工事士の資格を習得している。そのため、電気工事実習の単位は修得済みであるが、個別に日程調整等を行い、どの様な学生にでも対応できる指導体制を構築している。第二種電気工事士の試験は、前期にも上期試験が行われる。上期試験対策については、正課外で指導を希望する学生と指導を行う先生方のスケジュールを調整し、個別に平日の放課後や土曜日、日曜日に技能試験対策を実施することもある。上期試験を受験するケースとしては、前年度の下期試験で筆記試験には合格したが、技能試験で落ちてしまった学生が、翌年度の上期試験で筆記試験免除者として技能試験だけ受験することが多い。3、4 年生には、電気工事関連の企業への就職活動を考えて、事前に第二種電気工事士の勉強を始めて受験中であることをアピールしたい、もしくは、電気工事士の資格を取得している優位性を活かして就職希望先にアピールしたいなどと申し出があり、指導することもある。

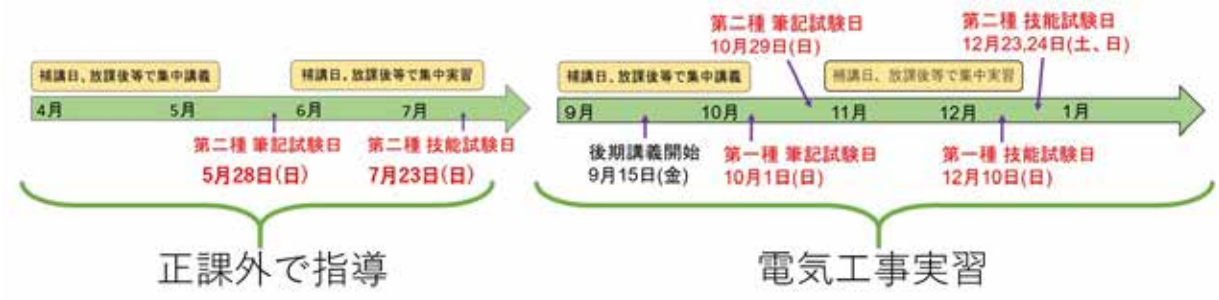


図 4 電気工事士資格取得支援スケジュール

3.2 筆記試験対策

現在までにどの様な形式で第二種電気工事士の筆記試験対策を実施しているのか紹介する。筆記試験対策では、電気回路の基本的な内容から復習し、電気工事士の試験で出題される基礎理論を解けるようにスライドや黒板を使った説明を行っている (図 5)。

図 5 筆記試験対策の様子

第二種電気工事士の筆記試験対策として有効な勉強方法の一つは、過去 10 年分の過去問を解き、覚えることである。なぜなら、電気工事士の筆記試験では、過去問と同じ問題が出題されることが多いためである。そのため、受講者には、10 年分の過去問と解説が載っているテキストを購入させて､授業前の事前学習として過去問を 1 年分解いてくることを課している。

授業時間内では、過去問を出題カテゴリ別にシャッフルしたオリジナル模擬試験問題を 2 回分解く (図 6)。その後、各自の理解度確認や苦手な問題を克服して合格ラインを超えるように、これまでの受講者の正答率が低かった問題を中心に作成した解説資料を配布して説明を行っている。

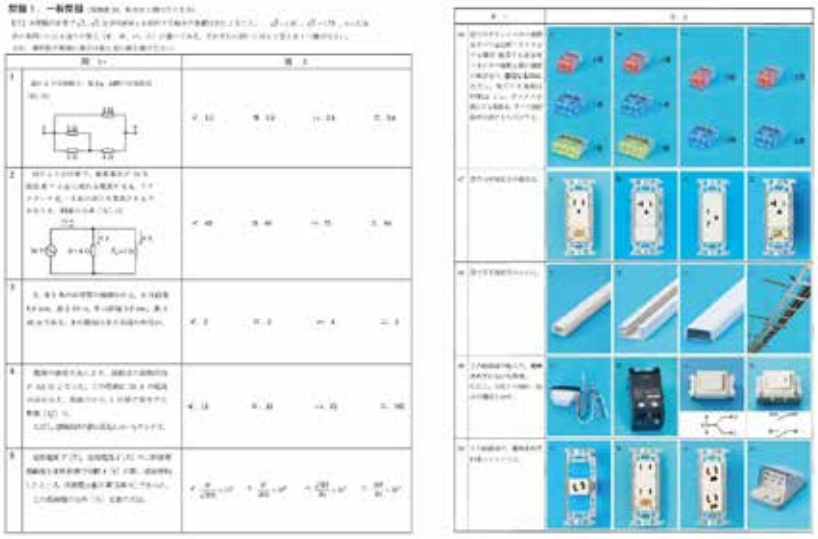


図 6 オリジナル第二種電気工事士の筆記試験対策模擬試験問題例

筆記試験では、20 問分の配線図問題が出題される。配線図問題の出題傾向としては難しい計算問題は無く、配線図に示されている図記号に関する問題や器具がどの様に接続されているのかを示す単線図から実際の電気配線のつながりを示す複線図への変換、ジョイントボックス内での電線の接続に関する問題が多く出題される。配線図問題は図記号や電気機器について名称等を暗記する必要がある。そのため、筆記試験対策では、初回で、図 7 に示す図記号に関する課題を出し、図記号と名称が一致するように指導している。図 7 では、名称が書き込まれているが、課題としては名称や図記号のどちらかを調べて書くようにしている。受講者は、模擬試験問題で出題された図記号の確認、暗記する際に利用している。

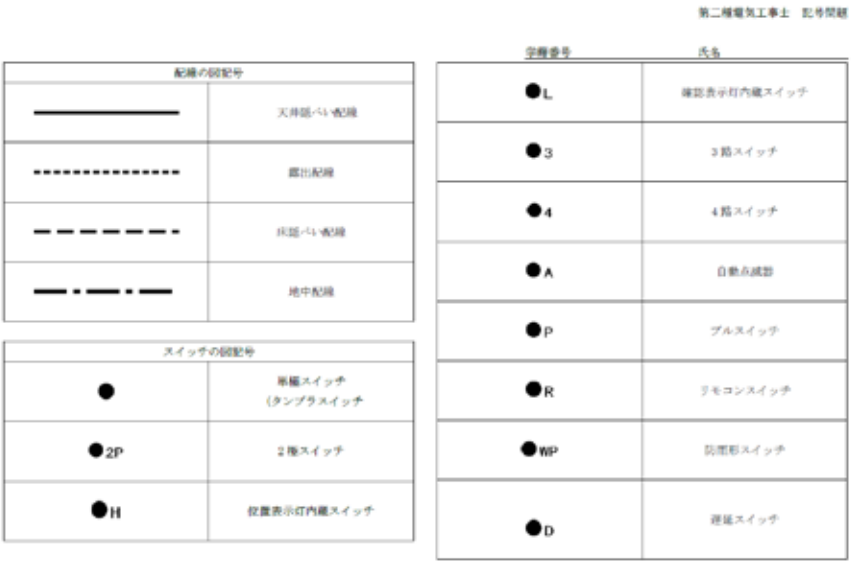


図 7 図記号に関する課題

単線図から複線図への変換手順については、全ての電路を複線図に直しているとミスを招きやすい。そのため、問題に応じた電路の切り分け作業が重要となる。電気工事士試験対策として販売されている教科書やアプリでは、簡単な単線図から複線図への変換手順については説明がなされている。一方で、実際に出題された配線図問題での効率的な複線図への変換手順である電路の切り分け作業は説明されていない。そのため、電路の切り分け作業の考え方についての解説資料を作成している (図 8)。各配線図問題で単線図から複線図へ変換が必要な例題を示し、手順を解説している。また、単線図から複線図への変換が上手くできるようになると問題によっては 5 問程度解答することができる。ゆえに、第二種電気工事士の筆記試験において合格ラインを超えるために押さえておきたい内容でもある。

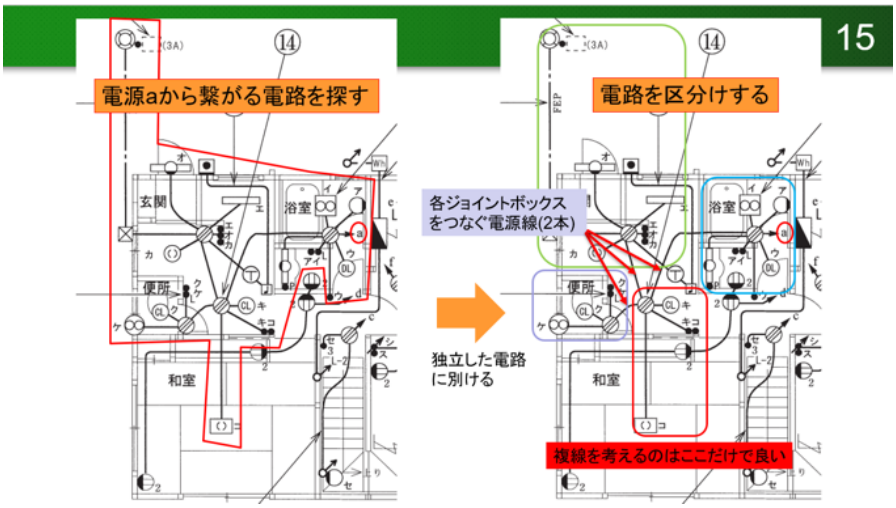


図 8 配線図問題に関する対策資料

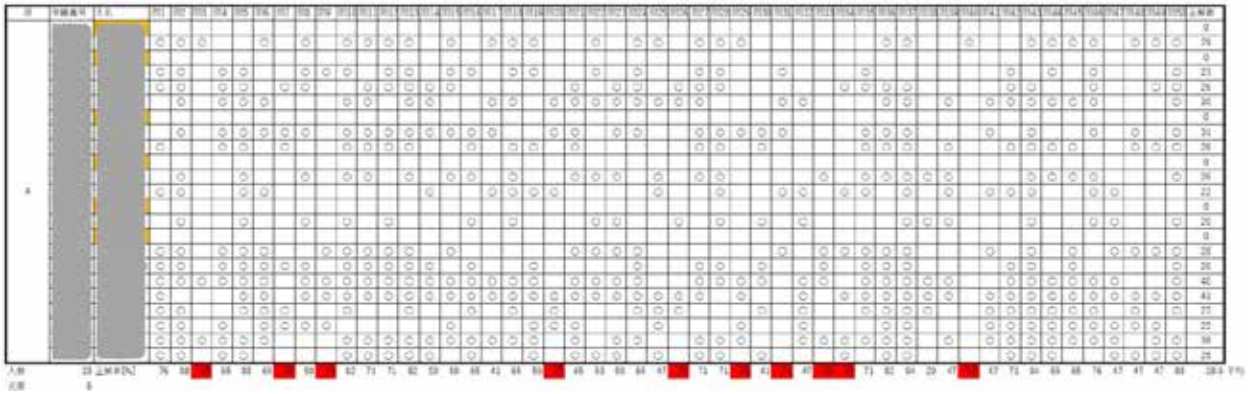


図 9 各学生の模擬試験問題結果の集計

次に、各学生の理解度の確認や全体で正解率の悪い問題を洗い出して次回解説するために、模擬試験問題結果の集計作業を行っている。筆記試験対策では、毎回、2 回分の模擬試験問題を出題している。回答用紙の採点は、学生自身で答えを見ながら行う。そして、授業内で回答用紙を回収して、授業後にエクセルで表を作り、各学生の回答用紙の正解について、問題 1 から 50 まで「○」を入れていく作業を行っている。個人の正解問題数や全体の平均正解問題数、各問題の正解率が自動的に計算される。模擬試験問題の難易度によって異なるが正解率が 3 割以下の問題について、次回の講義冒頭で解説きるように資料を準備している。ここで、集計作業は、受講生が 40 名近くになると、SA の学生と分担してエクセルに入力しても 2 時間ほどかかる。また、週に 2 コマ、多い時では土曜日 1 コマでも実施するため、集計作業が間に合わなくなりこともある。そのため、電気工事士の筆記試験対策アプリの機能として、自動集計ができ、各学生や全体の理解度や苦手とする問題の把握が簡便にできないかと考えている。現段階では、この機能に関しては実装するまでに至っていない。

3.3 技能試験対策

筆記試験に合格した学生を対象として、8 号館 5 階の電気工事実習専用の実習室で 12 月末の技能試験に向けて指導を行っている (図 10)。受講生には、1 人 1 個の工具セットを配布している (図 11)。この工具セットは、技能試験終了まで学生が自宅や空き時間などで作業ができるように貸し出している。また、技能試験の練習には、電線や器具が必要となる。電気工事実習では、学生の負担を無くし、十二分に技能試験の練習に取組めるように、様々な方法を用いて予算を確保している。その結果、近年では実習室に電線や器具を陳列する場所を作り、学生が必要に応じて電線や器具を取り、練習ができるようにしている (図 12)。

図 10 技能試験対策の様子

図 11 工具セット

技能試験では、あらかじめ公表問題として 13 課題が公表されている。技能試験対策の流れとしては、電気工事で使用する特殊な工具であるワイヤストリッパや圧着工具などの使い方を教えている。単純な工具であるが銅線を切断することができる工具であるため、学生がケガすることないように諸注意を行っている。初日の練習では、電球を取り付ける器具であるランプレセプタクルや引掛けシーリングなどに電線を接続するための技能や銅線同士を電気

的に接続する圧着工具を用いた作業を習得してもらう。特に、ランプレセプタクルの端子にVVF ケーブルの銅線で輪っかを作り、取り付ける作業は、不器用な学生は少し時間が掛かるが、何度も練習することで、切断する長さや輪っかを作るコツが掴める。図 13 に示す様に、実習では、貞方を含め梅崎技能員、森技能員、山光助手、前年度資格を取得した学生アルバイタで丁寧に指導している。また、学生アルバイタの指導では、昨年受験した経験を活かして、ちょっとした作業の時短テクニックを後輩に教えてくれていた。学生同士で指導することで、指導する立場の学生の技量や電気工事の理解が深まって学生自身の成長に繋がっていると感じている。

図 12 電線及び器具置き場

図 13 技能員と学生アルバイタが指導している様子

3.4 合格者の推移

表 1 に電気技術者センターが公開しているデータを基に作成した令和元年度からの令和 5 年度上期までの第二種電気工事士試験受験者数及び合格者数と合格率の推移についてまとめている [12]。令和 2 年度の上期については、コロナインフルエンザの影響で試験が中止になっているため、一部受験者数等のデータが無い。電気技術者センターが公表しているデータでは計算されていないが、各年の上期、下期での合格率を計算し掲載している。合格率の計算は次式 (1) で求めた。

$$合格率 = \frac{技能試験合格者数}{筆記試験受験者数} \times 100\ [\%] \qquad (1)$$

表 1 より、合格率は年度によって受験者数にばらつきはあるが、おおむね 5 割半ばである。

表 1 第二種電気工事士試験受験者数及び合格者数と合格率の推移

年度	期	受験申込者			筆記試験			技能試験			合格率[%]
		筆記申込者	筆記免除者	小計	申込者	受験者	合格者	申込者	受験者	合格者	
令和元年度	上期	84,529	9,142	93,671	84,529	75,066	53,026	62,168	58,699	39,585	52.7
	下期	54,794	17,548	72,342	54,794	47,200	27,599	45,147	41,680	25,935	54.9
	計	139,323	26,690	166,013	139,323	122,266	80,625	107,315	100,379	65,520	53.6
令和2年度	上期	—	7,236	7,236	—	—	—	7,236	6,884	4,666	—
	下期	121,951	5,102	127,053	121,951	104,883	65,114	70,216	66,113	48,202	46.0
	計	121,951	12,338	134,289	121,951	104,883	65,114	77,452	72,997	52,868	50.4
令和3年度	上期	95,351	17,447	112,798	95,351	86,418	52,176	69,577	64,443	47,841	55.4
	下期	79,274	14,571	93,845	79,274	70,135	40,464	55,035	51,833	36,843	52.5
	計	174,625	32,018	206,643	174,625	156,553	92,640	124,612	116,276	84,684	54.1
令和4年度	上期	88,008	11,981	99,989	88,008	78,634	45,734	57,715	53,558	39,771	50.6
	下期	75,728	12,714	88,442	75,728	66,454	35,445	48,159	44,101	31,117	46.8
	計	163,736	24,695	188,431	163,736	145,088	81,179	105,874	97,659	70,888	48.9
令和5年度	上期	78,546	10,603	89,149	78,546	70,414	42,187	52,790	49,547	36,250	51.5
	下期										
	計										

表 2 に、理工学部電気工学科の電気工事実習等を通じて指導した学生の合格者数の推移を示している。2018 年度から正課の授業として初めて、初年度は第一種電気工事士に合格した学生が 1 名、第二種電気工事士に合格した学生が 9 名であった。なお、( ) 内の数値は、電気工事実習の履修者でない学生、もしくは、上期に受験し合格した学生数の合計である。2 年目以降、徐々に合格者数が増加し 2020 年度以降、第二種電気工事士は 20 名以上の合格者数となった。また、第二種電気工事士を取得した学生が、次年度に、第一種電気工事士の試験を受けるケースが増えた結果、毎年 2 名程合格している。電気工事士の需要としては第一種電気工事士の方が高いため、今後は、第一種電気工事士まで取得したいと思う学生を増やし、指導していく必要がある。

表 2 電気工事実習での合格者数の推移

	2018	2019	2020	2021	2022
第一種	1(1)	0	2(2)	3(3)	2(2)
第二種	9(2)	13	22(4)	22(7)	22(10)

3.5 Advanced Program

2017 年度に電気工事実習という科目を新設した。しかし、当時、実習スペースはあるが、工具や電線、器具などがほとんどない状態であった。学科の予算内から電気工事実習用に予算を頂き、それでも足りないので研究室の予算などから捻出してスタートしていた。2 年目以降受講者が増えて、電気工事実習の予算だけでは、運営することが厳しくなってきた。そこで、学科外から予算を確保するために、KSU プロジェクト型教育として第一種及び第二種電気工事士の資格取得支援を申請した。その結果、2021 年度から理工学部 TopUp プログラム、KSU プロジェクト型教育として採択され予算を確保することに成功した。このお陰で、実習環境の構築は大きく進展し、受講者全員分の工具セットや電線と器具についても満足いく量を揃えることができるようになった。さらに、受験料の半額補助も行えるようになり、学生の金銭的な負担を低減させることができている。

現在は、九産大の教育大改革として、目的･目標を明確化し的確なプログラムで実践力育成、同じ夢をもつ仲間と一緒に切磋琢磨しながら夢を実現することをポイントに置いた Advanced Program が 2023 年度より新設されることになった (図 14)[13]。そのため、KSU プロジェクト型教育で取り組んでいた第一種及び第二種電気工事士の資格取得支援を Advanced Program に変更し新たにスタートしている。予算だけでなく、資格取得支援を行う指導者や学生アルバイタも増やすことができ指導体制がより強化されている。

数年に渡り、電気工事士の資格取得支援にご理解を頂き協力して頂いている学科教職員及び大学事務局等に感謝申し上げます。今後も学生が希望する将来に向けて一歩を踏み出せるように、様々な形で尽力させていただきます。

図 14 Advanced Program

4 Flutter

スマートフォンのアプリ開発は様々方法を用いることができる。その中で、2018 年にリリースされた Flutter は、Android や iOS のスマートフォンアプリ開発だけでなく、Windows などのデスクトップアプリや Web アプリ開発もこなせるクロスプラットフォーム開発環境としての機能性と将来性の高さから、Flutter を採用する事例が増えている [14]。本章では、クロスプラットフォーム開発の強み、Flutter とはどの様なクロスプラットフォームなのか説明する。Flutter で使われている言語である Dart について触れておく。また、Flutter の開発環境構築についての詳細は長くなるため、本家 Flutter サイトにある説明に任せ、大まかな流れを説明することとした。

4.1 クロスプラットフォーム開発

電気工事士の筆記試験対策アプリを開発するにあたり考えておかないといけないことがある。それは、どのプラットフォームで動作するアプリを作成するかである。授業での利用を考えると、理工学部の学生であれば Windows が走る貸与ノート PC を使える環境がある。もしくは、Mac book を使っている学生も居るかもしれない。また、学生だけでなく、だれでもスマートフォンを所持している時代である。各社スマートフォンメーカーが存在するが、スマートフォンのアプリが動作する OS を提供しているところは意外と少なく、Google か Apple である。Google が開発した「Android OS」をベースとした「Android」、Apple が開発した「iOS」がベースの「iPhone」の二強である。

アプリを各 OS 向けにネイティブアプリ開発を行うのであれば、XCode という開発環境が必要で、プログラミング言語は Objective-C や Swift。Android であれば Android Studio という開発環境で、プログラミング言語は Java や Kotlin を使う必要がある [15]。

ブラウザ (HTML5/Webkit) をベースとする「Web アプリ」として開発する方法で、アプリをブラウザ上で動作する「Web アプリ」として実装するモバイル端末のブラウザで動作させる手法である。しかし、この手法では、ブラウザのセキュリティ制約のために、各プラットフォームで利用可能な機能やハードウェアアクセスが制限されてしまう。そのため、比較的単純なアプリしか作成できない [16]。

上記の通り、学生が使用している PC やスマートフォンのプラットフォームに合わせて、いくつもの専用アプリを開発するのは、アプリ開発に馴染みのない初学者にとって大きなハードルとなる。世の中には、この様な状況を打破してくれる技術革新が起きている。それが、クロスプラットフォームやマルチプラットフォームと呼ばれる仕組みである。このキーワードは、昔から存在していたが、開発者の利便性などが全く異なる。iOS や Android、Windows や Mac など、デバイスや OS が異なる環境でも、同じ仕様のアプリを動かすことができるプログラムおよびその開発環境やフレームワークなどのツールがリリースされている。その一つが、今回使用している Flutter である。クロスプラットフォームを使うと、複数の OS 向けのアプリの大部分を、1 つの環境 1 つのプログラミング言語で開発することが可能になる。そのため、個々の OS に依存したプログラミングや開発環境についての内容を学習することがあまり必要でないため、学習コストが低くなる。

4.2 Flutter

Flutterとは、2017年にGoogleが発表したオープンソースのクロスプラットフォームの開発フレームワークである。Flutterはクロスプラットフォームなので、AndroidやiOS、Web、Windows、macOS、Linuxの6つのアプリを全く同じソースコードで開発できるのが魅力の一つである。

Flutter1では、モバイル(AndroidとiOS)のみが対象だったが、2021年のFlutter2でWebアプリ、2022年のFlutter3でWindows、macOS、Linuxのデスクトップアプリに正式対応し、モバイル以外のプラットフォームのアプリも同一のソースコードで開発できるようにバージョンアップが行われている。

4.3 Flutterの利点[15]

(1) 高い生産性

アプリ開発の生産性を考えると、iOSとAndroidの両プラットフォームを同一ソースコードから作成できることは個々のプラットフォームに向けて行う開発手法より優位である。さらに、個々のプラットフォーム向けのアプリをネイティブで開発することに比べても、次の理由で生産性が高いとされる。

① 表現力が豊かな言語と宣言的なアプローチで、より少ないコードでより多くのことができる
Dart言語で宣言的にUIを記述し、また、言語自体そのものに簡潔にUIが描けるように機能を追加されている。

② アプリの実行中にコードを変更してリロードすることを繰り返せる
ホットリロードは1秒以内で状態の廃棄もない。このため、気楽に実験できたり、UIや機能を追加できたり、バグを直ぐに修正できたりと、些細な変更も直ぐに動作確認ができる。

(2) 美しく高度なカスタマイズ可能なUI

アプリ作成で重要となるUIについて、デフォルトでも美しいMaterial DesignとiOS風のCupertinoウィジェット、豊富なモーションAPI、スムーズで自然なスクロール、そしてそれらは柔軟で表現力豊かにカスタマイズ可能である。スクロール、ナビゲーション、アイコン、フォントなど、独自にすべてが組込まれており、iOSとAndroidによるデザイン上の制限なしにカスタマイズできる。

(3) ネイティブパフォーマンス

Flutterでは、UIを独自にネイティブで組込んでいる。また、プログラムの実行前にコンパイルしておくAhead-Of-Time(AOT)コンパイルをサポートしている。このため、余計なブリッジ処理が挟まれることなく、ネイティブだけで実行可能になるので高速に動作可能である。

4.4 Flutter の開発環境構築

ここでは、簡単に Flutter の開発環境構築について示す。Flutter のインストールは下記のサイトから行える。

https://docs.flutter.dev/get-started/install

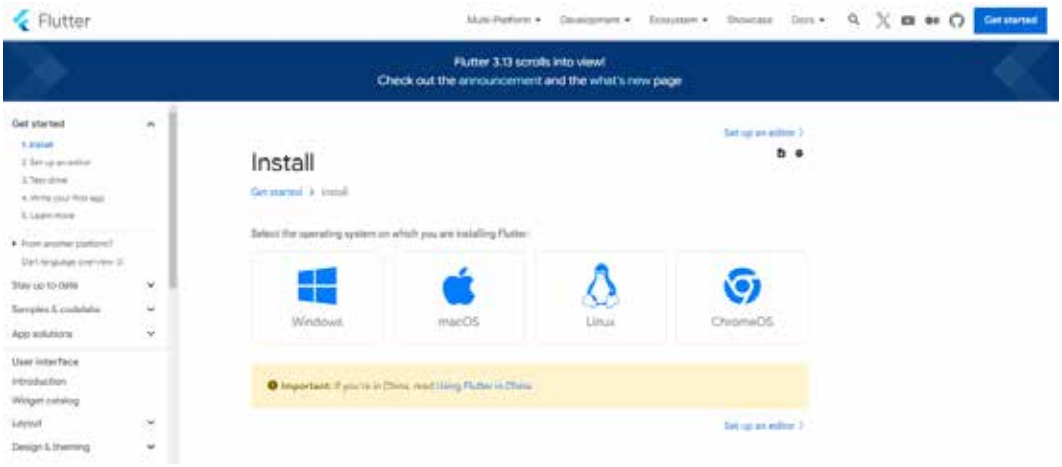


図 15 Flutter のインストール画面

図 15 に示す様に、開発に使用する PC の OS に合わせてインストールを進める。ここでは、Windows を選択して話を進める。Windows のマークをクリックすると図 16 のページが開く。System requirements にインストールするために必要な条件が書かれているので確認する。

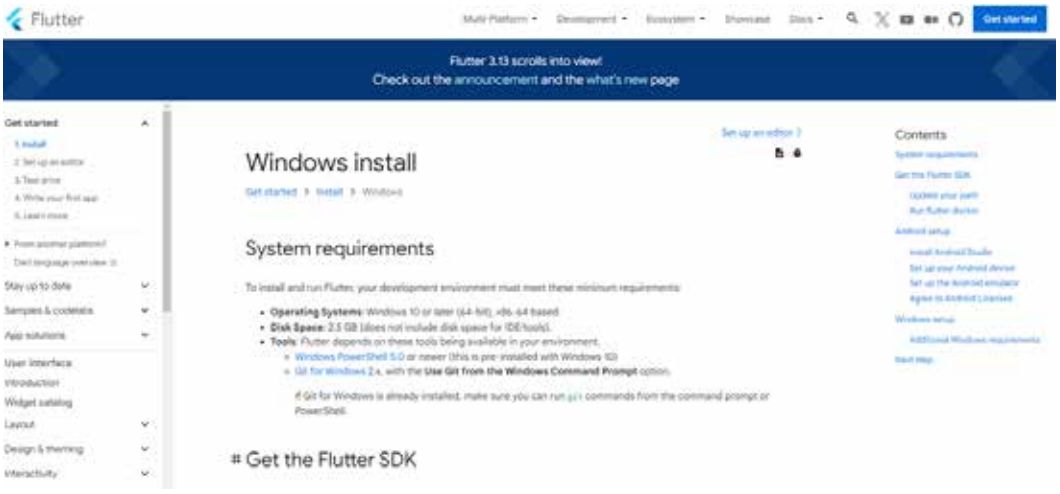


図 16 Windows install 画面

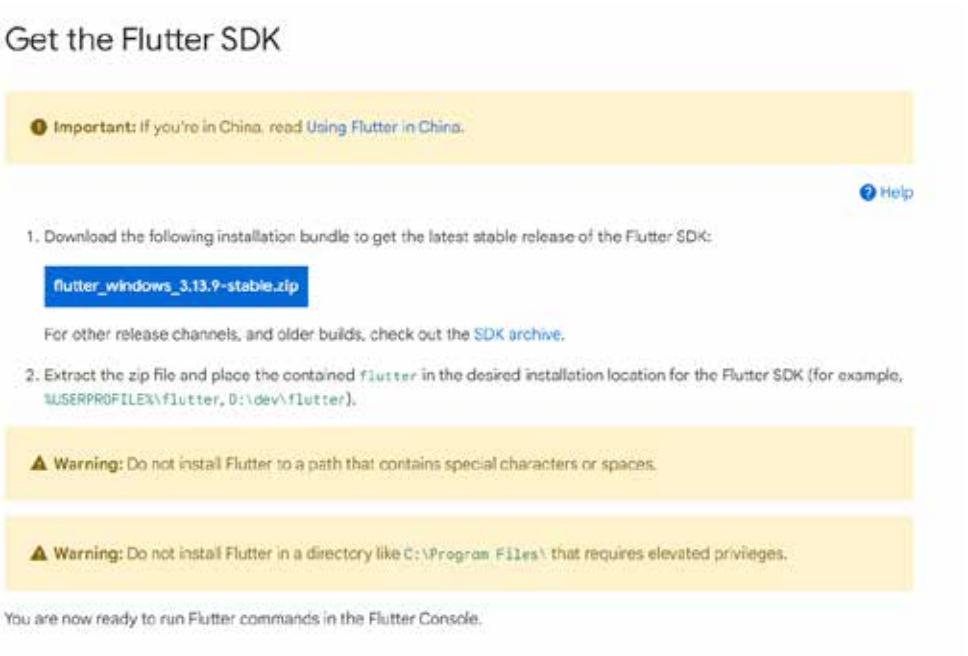


図 17 Get the Flutter SDK

インストール手順は図 17 に示す Flutter の SDK をダウンロードするところから始まる。サイトにかかれている手順通り行えば Android Studio を使った Flutter の開発環境を準備できる(図 18)。

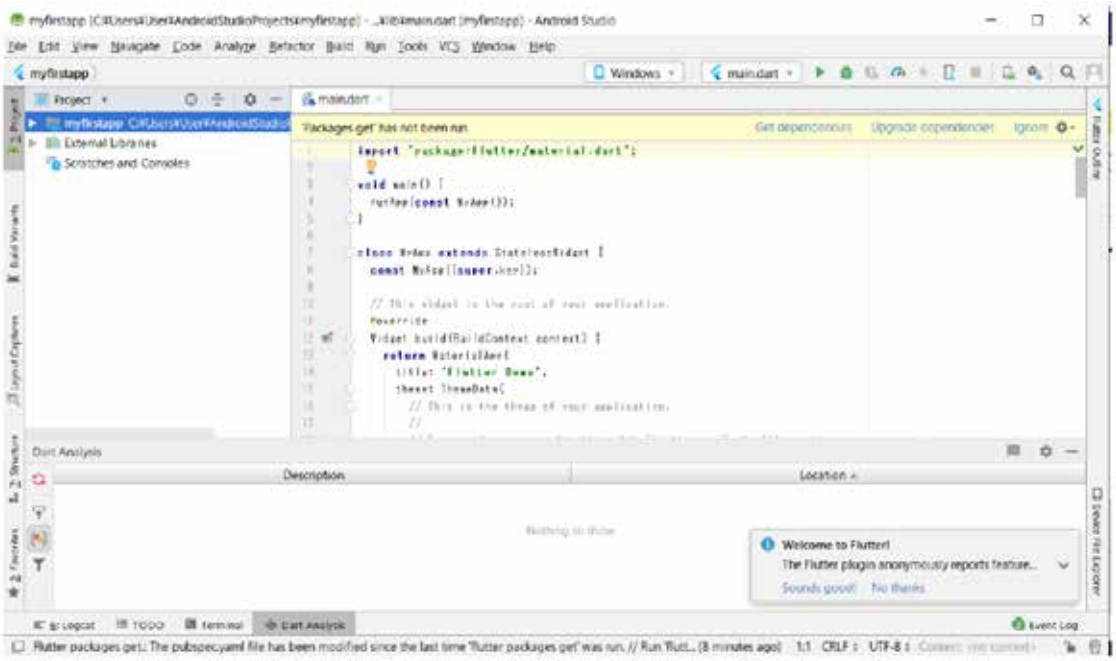


図 18 Android Studio

5 Flutter 入門

5.1 Firebase と連携させたメモアプリの作成例

(1) Firebase

モバイルアプリ開発を行っていると、データをスマートフォン内のリソースを介して読み書きするのではなく、クラウドにデータを蓄積し、各モバイルアプリからデータを利用する必要が生じる。今回の電気工事士筆記試験対策アプリでは、各学生が解いた過去問の回答結果などをクラウド上に保存し、指導する先生側のアプリで、クラウドのデータベースからデータを取得し表示することが考えられる。しかしながら､コンピュータやネットワークなどに精通していないとサーバーを立ててデータベースなどを構築し運用できるように準備するだけで、何度も日が暮れてしまう。昔ならそうであるが、現在では、データベースなどのサービスをクラウドで行ってくれる BaaS (backend as a service) と呼ばれるバックエンドサービスが利用できる。

本投稿では、Google が提供している BaaS の一種である Firebase と Flutter を連携させアプリを作成する方法を簡単に紹介する。Firebase は、スマートフォンアプリや Web アプリにおけるバックエンド開発において、スピードの向上とコスト削減を可能にするプラットフォームである。Firebase には、バックエンドを含め、アプリ開発に役立つ機能が数多く用意されている。しかも、基本は無料で使用できる。制限はあるが小規模なアプリでのデータベース使用であれば不都合なく利用でき便利である。

本節では、図 19 に示す単純なメモアプリの作成を通じて意外と簡単に Firebase が提供している Firestore のリアルタイム同期型のデータベースにメモ内容を追加、取出しができることを紹介する。

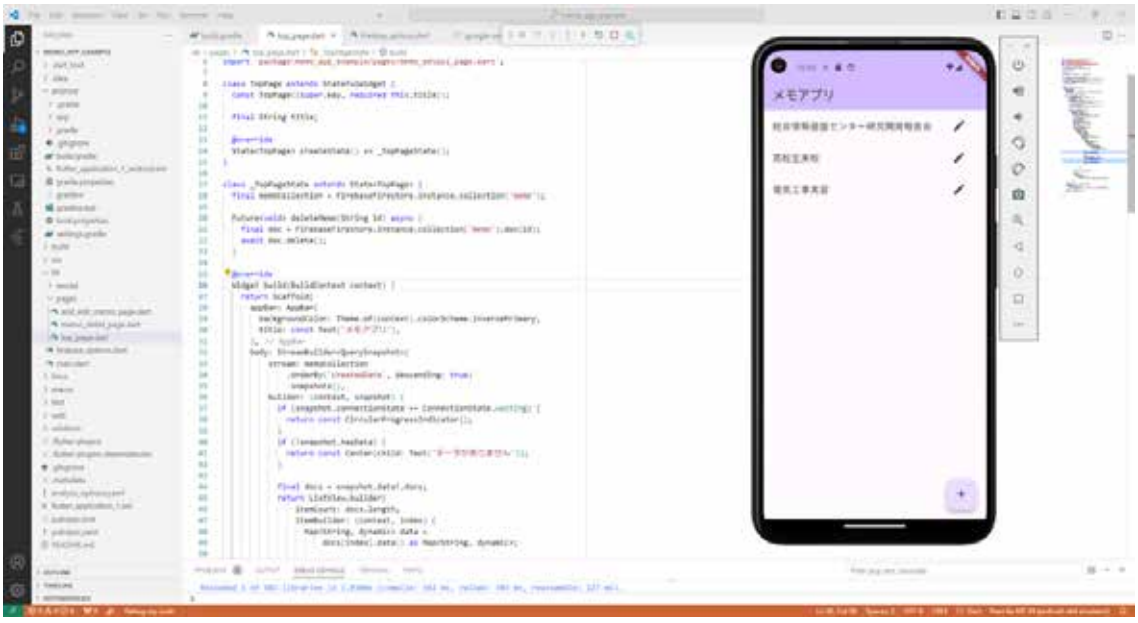


図 19 メモアプリの開発画面

Firebase の準備については、下記の本家 Firebase のサイト及び参考文献 [17] やネット上の記事を参照してください。ただし、書籍など古い情報だとバージョンアップでやり方が変わっていることがあるので注意が必要です。

https://firebase.google.com/?hl=ja

(2) メモアプリの動作紹介

図 20 メモアプリ動作 1

メモアプリの動作について紹介する。図 20(1) にアプリを起動したときのスタート画面を示す。この画面で、右下の十字マークのフローティングボタンをタップすると、図 20(2) のメモ追加画面に遷移する。ここでは、メモのタイトルと詳細を入力できる。入力後、追加ボタンをタップすることで図 20(3) に示すメモタイトルの表示画面に戻る。メモタイトルをタップすると、図 20(4) に示す様に、メモの詳細を確認することができる。

ここで、アプリから Firestore のデータベースに送られるデータについて紹介する。新規作成されたメモのデータは、Firestore のデータベースに新たなドキュメントとして保存される (図 21)。ドキュメントの中のフィールドに、作成日時 (createdDate)、メモタイトル (title)、詳細 (detail) が保存されていることがわかる。メモタイトルの表示や詳細の表示を行う際には、アプリ側から Firestore のデータベースにアクセスして、ドキュメントを取得している。

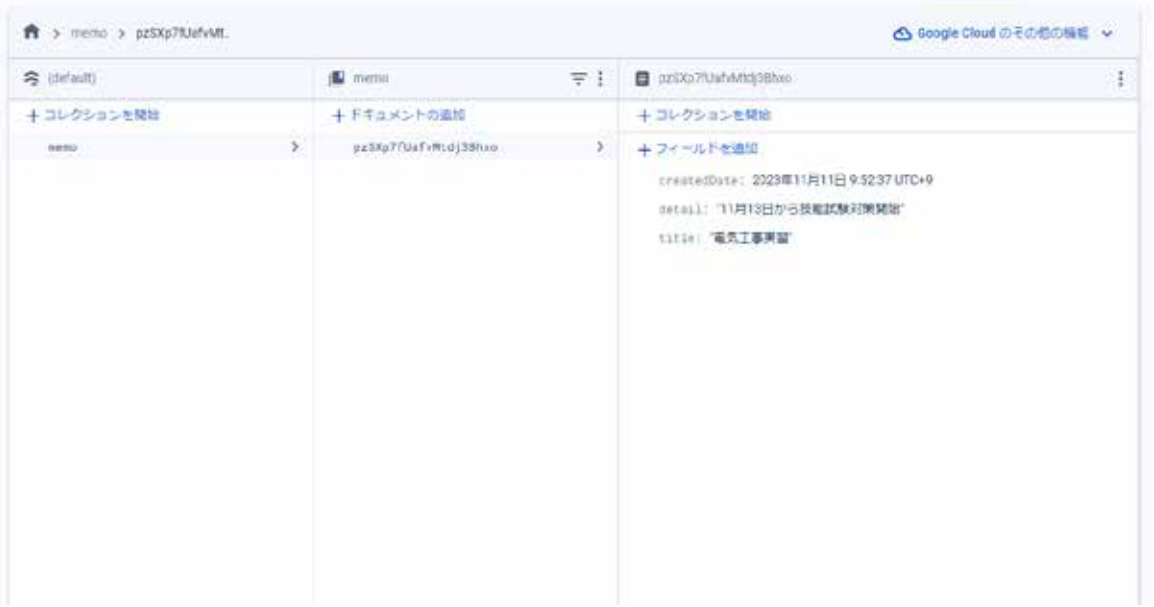


図 21 Firestore: memo コレクションへのドキュメント追加

図 22 メモアプリ動作 2

メモタイトルの右側の鉛筆マークをタップすると、図 22(1) に示す様な編集と削除のポップが出てくる。編集をタップすると、図 22(2) へ遷移して入力していた内容を確認し、修正ができる。図の例では、メモ詳細の内容を追加した。更新ボタンをタップすることでデータが Firestore の該当するドキュメントのフィールド値に書き込まれる。再度、メモタイトルをタップして、更新内容を確認すると変更されたメモ詳細を見ることができる (図 22(3))。

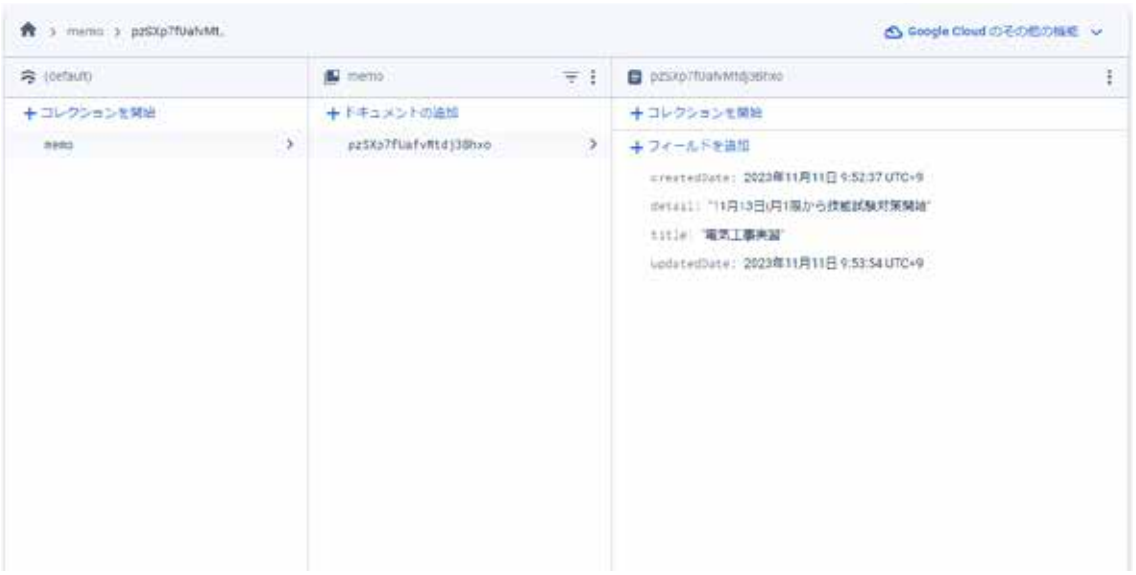


図 23 Firestore: memo コレクションのドキュメント変更結果

図 23にメモ詳細を更新したときのデータベースの違いを示す。更新すると詳細は勿論変更されているが、加えて、更新日時である updatedDate が追加されていることがわかる。この createdDate の値は、メモタイトルをタップし、編集画面へ遷移するときに使っている。createdDate の値が存在するときは、ボタンの表示を「メモ編集」とする。createdDate の値が無いときは、ボタンの表示を「メモ追加」とする。

(3) メモアプリのスタート画面のコード

図 24 に、初期画面へ遷移するために必要なプログラムを示す。1 と 3 行目で、Firebase で使用しているパッケージをインポートしている。6 行目の main 関数内で、Firebase を使用するための初期化処理 (Firebase.initializeApp) を記述している。そして、クラス MyApp 内でメモアプリの初期画面を作っている top_page.dart に記述している TopPage へ処理が移る。

```
lib > main.dart > ...
1  import 'package:firebase_core/firebase_core.dart';
2  import 'package:flutter/material.dart';
3  import 'package:memo_app_example/firebase_options.dart';
4  import 'package:memo_app_example/pages/top_page.dart';
5
   Run | Debug | Profile
6  void main() async {
7    WidgetsFlutterBinding.ensureInitialized();
8    await Firebase.initializeApp(
9      options: DefaultFirebaseOptions.currentPlatform,
10   );
11   runApp(const MyApp());
12 }
13
14 class MyApp extends StatelessWidget {
15   const MyApp({super.key});
16
17   // This widget is the root of your application.
18   @override
19   Widget build(BuildContext context) {
20     return MaterialApp(
21       title: 'Memo app',
22       theme: ThemeData(
23         colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
24         useMaterial3: true,
25       ), // ThemeData
26       home: const TopPage(title: 'Memo app'),
27     ); // MaterialApp
28   }
29 }
```

図 24 main.dart

図 25 メモリスト表示例

図 25 に、メモアプリに 2 つのメモを追加した画面を示す。この画面では、メモのタイトルのみが表示される。タイトルをタップするとメモの詳細が出てくる。右側の鉛筆マークをタップすると編集や削除ができる。

図 26 に main.dart から呼ばれた、TopPage クラスのプログラムを示す。17 行目から _TopPageState クラスが始まっている。その中で、18 行目に、Firestore データベースのコレクション：memo に保存されているドキュメントやフィールドの値を使用するための処理を書いている。

20 行目には、登録したメモ内容を削除する関数である。削除したいドキュメントの ID を渡すことでデータベースから削除できる。

32 行目からの body: で、StreamBuilder を用いている。StreamBuilder は、Flutter で非同期処理を行う際に役立つウィジェットである。基本的な役割は、指定されたストリームから流れてくるデータに応じて、自動的にウィジェットの再描画を行うこと。ゆえに、アプリケーションの状態が常に最新であることが保証される。

33 行目では、データベースの内容が変更されたときに呼び出される処理を記述している。並びをデータベースのメモを作成した時間を保存する createdDate を基に新しいメモが先頭に来るようにソートしている。
37 行目から 42 行目では、Firestore との接続待ち、データが登録されていないときは、画面中央に「データがありません」と表示するプログラムを書いている。

図 27 に、リスト形式でメモのタイトルを表示するためのデータ処理を書いている。リストに表示したメモタイトルの右側にある鉛筆マークをタッチすると、編集と削除の画面が下から出てくる処理を図 27 示すプログラムの 60 行目からの IconButton の記述内に入れている。

リスト表示したメモタイトルをタップすると、メモの詳細を表示する画面へ遷移する処理を 98 行の onTap で記述している。画面右下の十字マークのフローティングボタンをタップした時の処理を 107 行目から記述している (図 27)。

lib > pages > top_page.dart > _TopPageState

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
//import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:memo_app_example/model/memo.dart';
import 'package:memo_app_example/pages/add_edit_memo_page.dart';
import 'package:memo_app_example/pages/memo_detail_page.dart';

class TopPage extends StatefulWidget {
  const TopPage({super.key, required this.title});

  final String title;

  @override
  State<TopPage> createState() => _TopPageState();
}

class _TopPageState extends State<TopPage> {
  final memoCollection = FirebaseFirestore.instance.collection('memo');

  Future<void> deleteMemo(String id) async {
    final doc = FirebaseFirestore.instance.collection('memo').doc(id);
    await doc.delete();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: const Text('メモアプリ'),
      ), // AppBar
      body: StreamBuilder<QuerySnapshot>(
          stream: memoCollection
              .orderBy('createdDate', descending: true)
              .snapshots(),
          builder: (context, snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              return const CircularProgressIndicator();
            }
            if (!snapshot.hasData) {
              return const Center(child: Text('データがありません'));
            }
```

図 26 top_page.dart

```
58                  return ListTile(
59                    title: Text(fetchMemo.title),
60                    trailing: IconButton(
61                      onPressed: () {
62                        showModalBottomSheet(
63                            context: context,
64                            builder: (context) {
65                              return SafeArea(
66                                child: Column(
67                                  mainAxisSize: MainAxisSize.min,
68                                  children: [
69                                    ListTile(
70                                      onTap: () {
71                                        Navigator.pop(context);
72                                        Navigator.push(
73                                            context,
74                                            MaterialPageRoute(
75                                                builder: (context) =>
76                                                    AddEditMemoPage(
77                                                      currentMemo: fetchMemo,
78                                                    ))); // AddEditMemoPage // MaterialPageRoute
79                                      },
80                                      leading: Icon(Icons.edit),
81                                      title: const Text('編集'),
82                                    ), // ListTile
83                                    ListTile(
84                                      onTap: () async {
85                                        await deleteMemo(fetchMemo.id);
86                                        Navigator.pop(context);
87                                      },
88                                      leading: Icon(Icons.delete),
89                                      title: const Text('削除'),
90                                    ) // ListTile
91                                  ],
92                                ), // Column
93                              ); // SafeArea
94                            });
95                      },
96                      icon: const Icon(Icons.edit),
97                    ), // IconButton
98                    onTap: () {
99                      Navigator.push(
100                         context,
101                         MaterialPageRoute(
102                             builder: (context) => MemoDetailPage(fetchMemo))); // MaterialPageRoute
103                   },
104                 ); // ListTile
105               }); // ListView.builder
106         }), // StreamBuilder
107     floatingActionButton: FloatingActionButton(
108       onPressed: () {
109         Navigator.push(context,
110             MaterialPageRoute(builder: (context) => const AddEditMemoPage()));
111       },
112       tooltip: ('Increment'),
113       child: const Icon(Icons.add),
114     ), // FloatingActionButton
115   ); // Scaffold
116 }
117 }
118
```

図 27 top_page.dart の続き

(4) メモ追加や編集画面のコード

Firestore のデータベースへのメモデータの送信については、非同期処理で creatMemo と updateMemo 関数内で処理している (図 28)。

```
18  Future<void> creatMemo() async {
19    final memoCollection = FirebaseFirestore.instance.collection('memo');
20    memoCollection.add({
21      'title': titleController.text,
22      'detail': detailController.text,
23      'createdDate': Timestamp.now(),
24    });
25  }
26
27  Future<void> updateMemo() async {
28    final doc = FirebaseFirestore.instance
29        .collection('memo')
30        .doc(widget.currentMemo!.id);
31    await doc.update({
32      'title': titleController.text,
33      'detail': detailController.text,
34      'updatedDate': Timestamp.now()
35    });
36  }
```

図 28 add_edit_memo_page.dart

6 第二種電気工事士筆記試験対策アプリケーション開発

本章では、第二種電気工事士の資格取得支援のために試作しているスマートフォンアプリ開発について説明する。電気工事士の試験は筆記試験と技能試験に別れているが、今回は、筆記試験を対象としている。

6.1 第二種電気工事士筆記試験対策アプリケーションの仕様

電気工事士筆記試験対策アプリとして必要な機能を考えると、教員側と学生側で求められる内容が異なってくる。

(1) 学生の立場で必要な機能等

1. 過去問を解く機能

電気工事士の筆記試験では、過去問がそのまま出題されることが多いため、過去問を繰り返し解き、間違えた問題を覚えていくことが重要である。学生は、過去 10 年間分の過去問と解説、理論などの説明が載っているテキストを用いて過去問を解き繰り返し自習している。何度か解いていくと、選択肢の記号や解答を覚えてしまって、あまり良い練習にならない例が起こっている。そのため、電気工事実習では、そのようなことが起きないように、模擬試験問題に出題する問題はシャッフルしたものを毎回準備している。シャッフルした模擬試験問題にチャレンジすることで実力を試すことができる。そのため、試験対策アプリの機能として、過去問をそのまま解く機能とシャッフルした過去問を解く機能が必要となる。

2. 解説機能

問題を解いた後に、その解説まで見れるようにする必要がある。スマートフォンアプリとして電気工事士のアプリがリリースされているが、問題は出題されても、その解説は無い場合が多い。空き時間などでスマートフォンだけで問題を解いて勉強したいと思うと効率的ではない。そのため、各問題の答えと解説を見れる機能は必要となる。

3. 問題カテゴリごとの勉強機能

第二種電気工事士の筆記試験では、下記に示す(ア)から(キ)までのカテゴリの問題が出題される。この中には(ア)や(イ)のように基礎理論等で計算が必要になる問題がある。計算が苦手な学生は、できるだけ計算が解けるようにこの部分を集中的に練習すると良い。また、(キ)の法令などは、細かい数値を覚える必要があるため間違いやすい。そのため、テキストに載っている過去問から同じカテゴリの問題を見つけ出し解いてみるしかない。しかし、かなり手間であるので、最初から問題カテゴリ別に回答することができる機能があると、苦手な分野を繰り返し勉強し、合格ラインを超えることができると考えている。

問題カテゴリ

(ア) 電気に関する基礎理論
(イ) 配電理論及び配線設計
(ウ) 電気機器・配線器具並びに電気工事用の材料及び工具
(エ) 電気工事の施工方法
(オ) 一般用電気工作物等の検査方法
(カ) 配線図
(キ) 一般用電気工作物等の保安に関する法令

4. 図記号暗記問題機能

配線図では、コンセントやスイッチなどの図記号を覚えておかないと回答できない問題が多く出題される。英単語を覚えるための暗記カードの様に、図記号を覚えることができる機能があると空き時間にチラチラと見て覚えることができる。

5. 写真鑑別問題機能

工具や器具などの写真が出題され、名称を回答、その逆の出題がある。これに関しても、図記号と同様な機能を実装することで回答率を高めることができる。

6. 問題カテゴリごとのまとめ資料

上記の問題カテゴリごとの解説資料を見て、計算の公式などを確認できる機能。

7. 過去に解いた問題のふり返り機能

過去問などどこを間違えていたのかなどを確かめることができる機能。また、各問題の正解率がわかることで、どこが苦手なのか把握することができる。

8. 質問機能

何か聞きたいことがあったときに、メールを送るのは面倒だと思う学生がいる。そのため、質問を手軽に送れるようにすると学生の理解向上につながる。

9. 他学生の学習状況の共有機能

匿名で、他学生がどのくらい過去問を回答しているのか、点数、正解率がどうなのかがわかる機能。これは、1 人だけで学習するよりも、少し他の学生と競い合うことで、学習のモチベーションを高めていける。

(2) 教員の立場で必要な機能等

1. 学修管理機能

試験対策アプリを使用する学生がログインするため必要な ID やパスワード、連絡先のメールアドレスを登録できる機能。また、各学生が、どのくらいの頻度で勉強しているのか、正解率や不得意な問題は何かを調べることができる機能が必要である。PC でブラウザなどから確認できると良い。

2. 簡単な通知機能

ログイン画面に連絡通知が表示される枠を設けることで、授業等の連絡を一貫して行える。

3. 問題等の更新機能

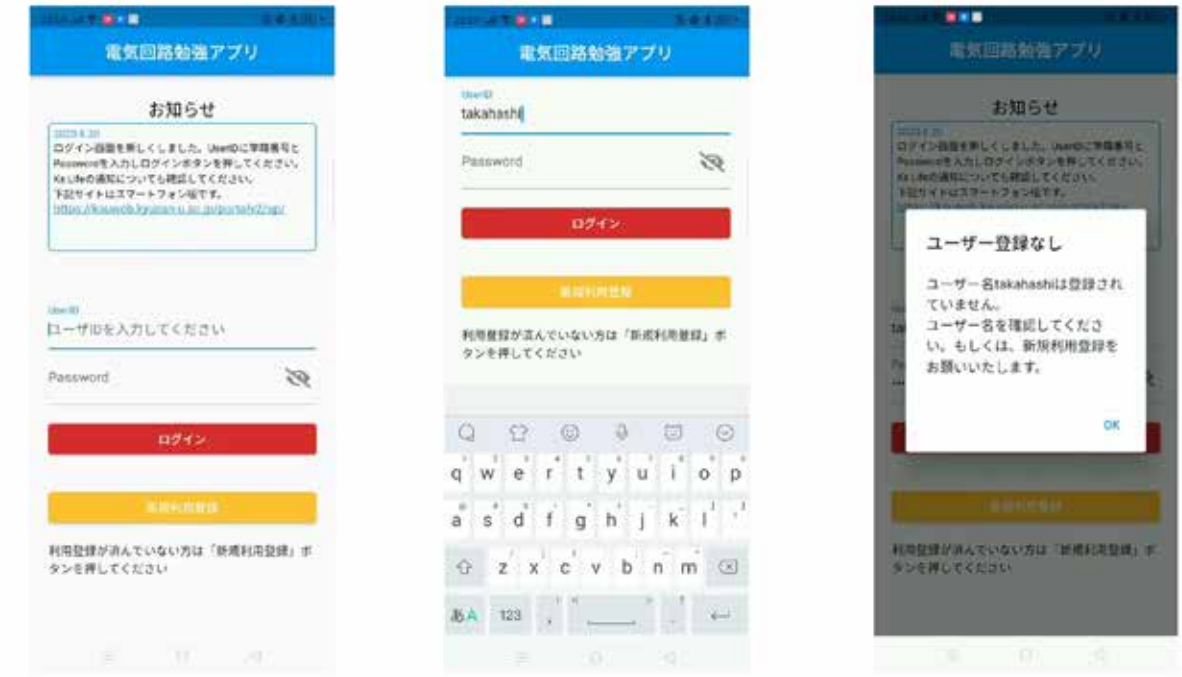
スマートフォンからではなく、PC のブラウザ経由で過去問の追加や新しい問題の追加などが手軽にできると良い。

上記の内容を踏まえて、電気工事士の筆記試験対策アプリの開発に取り組んだ。ただし、全ての機能を実装できているわけでは無い。この点は、今後の課題である。

6.2 第二種電気工事士筆記試験対策アプリケーション開発状況の紹介

第二種電気工事士筆記試験対策アプリとして、試行錯誤しながら様々な機能を作ってきた。ここでは、各機能について、画面とプログラム抜粋を示しながら説明を行う。

(1) ユーザー登録及びログイン画面



(1) ログイン画面　(2) ユーザーID等入力　(3) 登録なしの場合

図 29 ログイン画面

図 29 に第二種電気工事士筆記試験対策アプリのログイン画面を示す。同図 (1) のログイン画面には、お知らせ欄やユーザー ID とパスワード入力フォーム、ログインボタン、新規利用登録ボタンを設けている。お知らせ欄については、あとで説明を行う。ここでは、ログイン機能に関して説明する。同図 (2) で示す様に、ユーザー ID「takahashi」とパスワードを入力して、ログインボタンを押す。この時、ユーザーデータベースに登録されていないユーザー ID「takahashi」の場合は、同図 (3) に示す様にユーザー ID が登録されていないことをユーザーにお知らせする。OK ボタンを押すと、元のログイン画面に遷移する。なお、ユーザー ID が正しく、パスワードが間違っている場合でも同様のメッセージが表示される。

図 30 ログイン画面からスタート画面への遷移

ログイン画面の新規利用登録ボタンを押すと、図 30(1) に示す新規利用登録画面が開く。ここでは、ユーザー ID として学籍番号 (例では名前になっている)、パスワード (6 文字以上) を確認のため 2 回入力して頂く。そして、新規登録ボタンを押す。もし、既に登録しているユーザー ID やパスワードが異なる時には同図 (2) に示すメッセージが表示される。ユーザー ID とパスワードに間違いが無ければ、同図 (3) に示すアプリのスタート画面が表示される。

次に、ログイン機能に関してのプログラムを説明する。ログイン画面を構成する全体のプログラムは 300 行ほどあるため、ここでは、ユーザー ID とパスワードを入力し、ログインボタンを押して、ユーザー ID がデータベースに登録されているのか確認する手順を説明する。なお、開発当時のユーザーデータベースについては、クラウドの Firestore ではなく、スマートフォン内に構築している。ユーザー情報を記録するデータベースとして Sqflite を用いた。ログイン機能に関するプログラムを図 31 に示す。194 行目でユーザー ID フォームに入力された ID がユーザーデータベースに登録されているか確認している。なお、プログラム中の print 分はスマートフォンの画面に表示されるものではなく、デバック用に VSCode のターミナルに表示させるためである。もし、登録されていない場合は、200 行目のユーザー登録なしを示すダイアログを表示する処理が動き始める。ユーザー登録がある場合は、220 行目からの処理が実行され、スタート画面へ遷移するようになっている。この時に、ユーザーデータとして、ログイン回数やログイン時間を更新している。

```
padding: const EdgeInsets.symmetric(vertical: 16),
child: ElevatedButton(
  style: ElevatedButton.styleFrom(
    primary: Colors.red[700], //ボタンの背景色
  ),
  onPressed: () async {
    //context.read<LogInModel>().setMessage('');

    if (_formKey.currentState.validate()) {
      _formKey.currentState.save();

      //データベースに登録済みか確認
      print('ユーザー名${UserID}がデータベースに登録されているか確認');
      if (await database.searchUserId(UserID) == false) {
        //ユーザーが登録されていない場合
        print('ユーザー名${UserID}は該当なし');

        // ダイアログを表示----------------------------------
        var result = await showDialog<int>(
          context: context,
          barrierDismissible: false,
          builder: (BuildContext context) {
            return AlertDialog(
              title: Text('ユーザー登録なし'),
              content: Text(
                  'ユーザー名${UserID}は登録されていません。\nユーザー名を確認してください。もしくは、新規利用登録をお願いいたします。'), // Text
              actions: <Widget>[
                TextButton(
                  child: Text('OK'),
                  onPressed: () =>
                      Navigator.of(context).pop(1),
                ), // TextButton
              ], // <Widget>[]
            ); // AlertDialog
          },
        );
      } else {
        //ユーザー名が登録されている場合
        ///パスワードを確認
        if (await database.getPassword(UserID) ==
            _password) {
          //ユーザー名とパスワード認証完了。スタート画面へ遷移

          //ユーザーidの取得
          final id = await database.getId(UserID);

          //ログイン回数を更新
          database.userDataList[id].loginTimes += 1;
```

図 31 ログイン関係のプログラム

```
Container(
  // margin: EdgeInsets.fromLTRB(0, 8, 0, 0),
  width: double.infinity,
  child: Padding(
    padding: const EdgeInsets.symmetric(vertical: 16),
    child: ElevatedButton(
      style: ElevatedButton.styleFrom(
        primary: Colors.yellow[700], //ボタンの背景色
      ),
      onPressed: () async {
        UserID = await Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => Registration(),
          ), // MaterialPageRoute
        );
      },
      child: const Text('新規利用登録'),
    ), // ElevatedButton
  ), // Padding
), // Container
Container(
  child: Text("利用登録が済んでいない方は「新規利用登録」ボタンを押してください"),
), // Container
```

図 32 新規利用登録画面へ遷移するプログラム

図 32 に、新規利用登録ボタンが押された場合の処理を示す。297 行目で、ボタンが押されると MaterialPageRoute を使って新規利用登録を行う画面へ遷移するプログラムとなっている。

図 33 に、新規利用登録画面のユーザー ID 及びメールアドレス入力フォームのプログラムを示している。メールアドレスの入力チェックとして、空欄や @ マークが無いなどの確認を 128 行目で行っている。プログラムは示していないが、パスワード入力フォームの確認処理では、入力された文字列が 6 文字以上か確認し、また、2 個入力したパスワードが同じであるか確認している。そして、ユーザーデータベースに登録されていないユーザーであれば、ユーザーデータベースに新規登録される。

```
98                TextFormField(
99                  controller: _textController_userId,
100                 decoration: const InputDecoration(
101                   labelText: 'UserID',
102                   hintText: 'ユーザIDを入力してください',
103                 ), // InputDecoration
104                 //autovalidate:
105                 //false, // 入力変化しても自動でチェックしない。
106                 // validator: context.read<LoginModel>().emptyValidator,
107                 validator: (value) {
108                   // _formKey.currentState.validate()でコールされる
109                   if (value.isEmpty) {
110                     return 'ユーザー名を入力してください'; // エラー表示のメッセージを返す
111                   }
112                   UserID = value;
113                   return null; // 問題ない場合はnullを返す
114                 },
115                 onSaved: (value) => () {
116                   // this._formKey.currentState.save()でコールされる
117                   print('$value');
118                 },
119               ), // TextFormField
120               TextFormField(
121                 controller: _textController_Email,
122                 decoration: const InputDecoration(
123                   labelText: 'E-Mail address',
124                   hintText: 'メールアドレスを入力してください',
125                 ), // InputDecoration
126                 validator: (value) {
127                   // _formKey.currentState.validate()でコールされる
128                   if (value.isEmpty || !EmailValidator.validate(value)) {
129                     return '正しいメールアドレスか確かめてください'; // エラー表示のメッセージを返す
130                   }
131                   email = value;
132                   return null; // 問題ない場合はnullを返す
133                 },
134                 onSaved: (value) => () {
135                   // this._formKey.currentState.save()でコールされる
136                   print('$value');
137                 },
138               ), // TextFormField
```

図 33 新規利用登録のプログラム

(2) スタート画面

図 34 に第二種電気工事士筆記試験対策アプリのメイン画面となるスタート画面を示す。スタート画面では、ユーザー ID をトップに示し、ログイン回数、前回ログインからの経過時間を示す様にしている。電気回路に関する基礎的な問題等を選択して練習することができるように、各メニューをリスト形式で表示できるようになっている。画面では、途中までしか表示されていないが、画面をスクロールさせることで下部に隠れているメニューも見ることができる。各メニューの構成は、タイトル、チャレンジボタン、☆マークとなっている。チャレンジボタンを押すことで各問題に回答する画面に遷移する。☆マークは、各問題をどのくらい回答したかを表す。5 回で一つ☆マークが付くようにしている。画面の下部には質問アイコンや、講義アイコン、チャレンジログアイコンを設置している。これらのアイコンについての説明は後ほど行う。

図 34 スタート画面

次に、スタート画面の主要なプログラムについて紹介する。図 35 に、スタート画面の上部に表示しているユーザー情報を取得更新するプログラムを示している。ユーザーデータベースからユーザー ID を基に前回のログイン時間を取得し、現在時間との差を計算することで経過時間を出している。図 36 に示すプログラムで、実際にユーザー情報を画面上に出力している。

```dart
class _UserMenue extends State<UserMenue> {
  _UserMenue(this.id, this.database, this.ansLogDatabase) {
//ラストログイン日時の取得
    DateFormat outputFormat = DateFormat('M月d日H時m分');
    String lastlogindate = outputFormat
        .format(DateTime.parse(database.userDataList[id].lastLoginTime));
    //前回ログイン時間からの時差を出す
    DateTime now_date = DateTime.now();
    diff_time = now_date
        .difference(DateTime.parse(database.userDataList[id].lastLoginTime))
        .inSeconds;

    hour = (diff_time / (60 * 60)).floor();
    mod = diff_time % (60 * 60);
    minutes = (mod / 60).floor();
    second = mod % 60;

    //ラストログイン時間を更新
    String now = DateTime.now().toString();
    database.userDataList[id].lastLoginTime = now;

    //データベースの更新
    database.updateSharedPrefrences();
  }
```

図 35 スタート画面のユーザー情報の更新等

```dart
  return new MaterialApp(
    debugShowCheckedModeBanner: false,
    title: 'スタート画面',
    home: new Scaffold(
      appBar: new AppBar(
        centerTitle: true, // 中央寄せを設定
        title: new Text('スタート画面'),
      ), // AppBar
      body: Column(
        children: <Widget>[
          Container(
            child: Center(
              child: Text(
                // 'ようこそ${database.userDataList[id].userId}さん\n' +
                //     '${database.userDataList[id].loginTimes}回目のログイン!\n' +
                //     '前回ログインは$lastlogindate',
                hour != 0
                    ? 'ようこそ${database.userDataList[id].userId}さん\n' +
                        '${database.userDataList[id].loginTimes}回目のログイン!\n' +
                        '前回から${hour}時間${minutes}分振りですね'
                    : 'ようこそ${database.userDataList[id].userId}さん\n' +
                        '${database.userDataList[id].loginTimes}回目のログイン!\n' +
                        '前回から${minutes}分振りですね',

                style: TextStyle(
                  fontSize: 20.0, // 文字サイズ
                  fontWeight: FontWeight.w500, // 文字の太さ
                  color: Colors.black, // 文字の色
                  //letterSpacing: 3.0, // 文字と文字のスペース
                  //FontFamily: 'Lato', // フォントの種類
                ), // TextStyle
                textAlign: TextAlign.center,
              ), // Text
            ), // Center
          ), // Container
```

図 36 スタート画面のユーザー情報の表示処理

図 37 に、スタート画面に表示している問題メニュー表示処理部を示している。各問題メニューの記述は、163 行目から始まる SingleChildScrolView 内に記述しているため、メニュー全体の縦の長さが画面サイズを超えた場合、スクロールが働くようになっている。問題メニューの処理については、

```
_buildQuestionMenue(' 合成抵抗計算 ', 20.0, Q_Star[0])
```

という関数を作成しており、そちらで問題タイトル表示、チャレンジボタンの表示、☆マークの表示処理を担当させている。そちらのプログラムの掲載は、かなり長くなるので割愛させていただく。

```
162             Expanded(
163               child: SingleChildScrollView(
164                 scrollDirection: Axis.vertical,
165                 child: Column(
166                   crossAxisAlignment: CrossAxisAlignment.center,
167                   children: <Widget>[
168                     //問題 合成抵抗計算
169                     _buildQuestionMenue('合成抵抗計算', 20.0, Q_Star[0]),
170                     SizedBox(
171                       height: 10,
172                     ), // SizedBox
173                     //問題 分圧計算
174                     _buildQuestionMenue('分圧計算', 20.0, Q_Star[1]),
175                     SizedBox(
176                       height: 10,
177                     ), // SizedBox
178                     //問題 分流計算
179                     _buildQuestionMenue('分流計算', 20.0, Q_Star[2]),
180                     SizedBox(
181                       height: 10,
182                     ), // SizedBox
183                     //問題 電力計算
184                     _buildQuestionMenue('電力計算', 20.0, Q_Star[3]),
185                     SizedBox(
186                       height: 10,
187                     ), // SizedBox
```

図 37 問題メニュー表示処理

(3) 合成抵抗計算問題

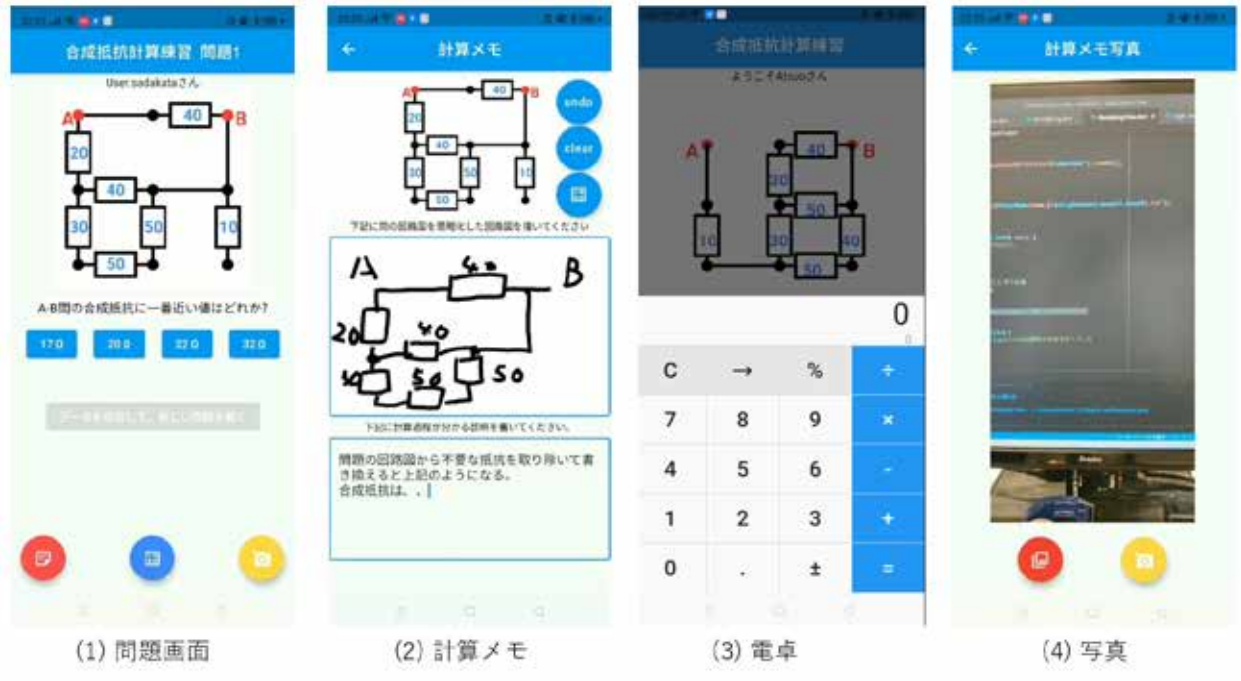


図 38 合成抵抗計算問題の各画面

第二種電気工事士の筆記試験で、必ず出題される問題である合成抵抗の計算を練習することができる機能について説明する。図 38 に合成抵抗計算問題の画面を示す。この合成抵抗計算では、図 39(1) に示す様な、田んぼの田の字の各辺に抵抗が配置された回路を考える。そして、A と B 間の合成抵抗を計算する。なお、この回路の各抵抗の有無や抵抗値は毎回ランダムに決まるようにしている。選択肢は 4 択としている。選択肢の値は、正解を含む、正解と大きな差が無い適当な値を表示するようにしている。

画面下に赤ボタンを押すと同図 (2) の計算メモ画面が開く。この計算メモでは、出題された回路について、どの様な過程で合成抵抗を求めたのかわかるように、指で回路図を描いたりすることができるお絵描き機能を入れている。また、回答について記述できるように入力フォームも設けている。

計算過程で手計算では難しいともあるかと思い、同図 (3) に示す様に、電卓が使えるようになっている。電卓は問題画面の下の青いボタンを押すと利用できる。

さらに、レポート用紙などに、回路図や式などをまとめて書きながら計算したい学生のために、レポート用紙をカメラで撮影、写真データを提出できる機能を実装している (同図 (4))。

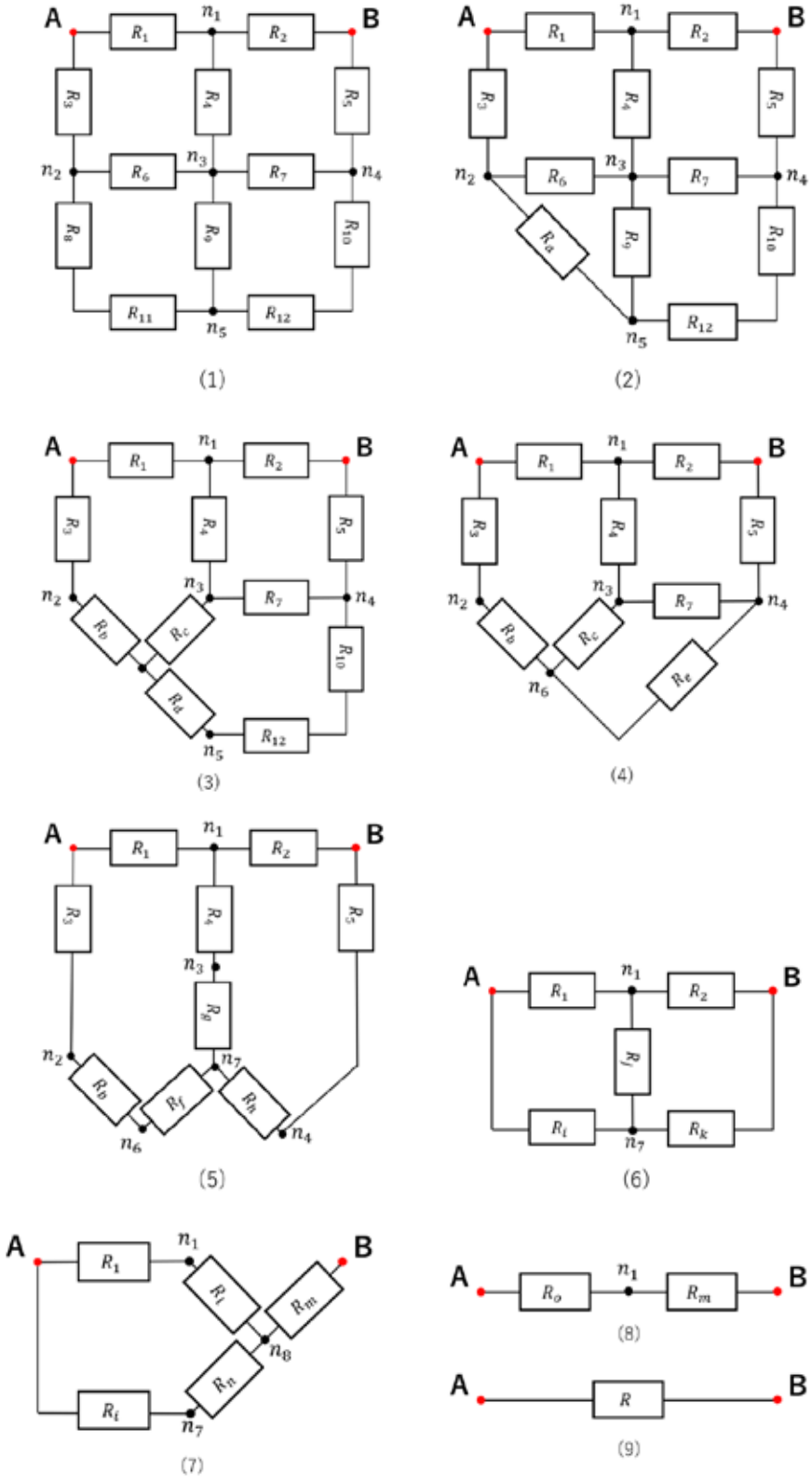


図 39 合成抵抗値の計算

次に、図 39(1) に示す回路の合成抵抗を求める過程について説明する。今回作成した合成抵抗を計算するアルゴリズムは、非常に単純で、全ての抵抗が存在する場合の合成抵抗を計算する式を予め解いておく。そして、問題出題時にランダムに抵抗の有無や抵抗値を決め、合成抵抗の式に代入することで機械的に求めることができる。

合成抵抗の計算は、複雑そうであるが、スターデルタ変換という電気工学科の 1 年生が知っているレベルの計算で解くことができる。しかし、解きたいと思うかは別である。それでは、計算について同図 (1) から (3) の変形を説明する。ここでは、節点 n_2, n_3, n_5 間の抵抗 R_6, R_9, R_8, R_{11} がデルタ接続になっている。

$$R_a = R_8 + R_{11} \quad (2)$$

ここで、式 (2) に示す様に、二つの抵抗をまとめると、同図 (2) の回路となる。

節点 n_2, n_3, n_5 間の抵抗 R_6, R_9, R_a のデルタ接続を同図 (3) に示すスター接続に変換すると各抵抗 R_b, R_c, R_d は下記の通りになる。

$$R_b = \frac{R_6 R_a}{R_6+R_9+R_a},\ R_c = \frac{R_6 R_9}{R_6+R_9+R_a},\ R_d = \frac{R_9 R_a}{R_6+R_9+R_a} \quad (3)$$

同図 (4) への変形は、節点 n_4, n_5, 間の抵抗を足すだけである (式 4)。

$$R_e = R_{10} + R_{12} + R_e \quad (4)$$

同図 (5) への変形は、同図 (4) の節点 n_3, n_4, n_6 間の抵抗 R_7, R_c, R_e がデルタ接続になっている。この部分をスター接続に変形すると各抵抗 R_f, R_g, R_h は下記の通りになる (式 5)。

$$R_f = \frac{R_c R_e}{R_7+R_c+R_e},\ R_g = \frac{R_7 R_c}{R_7+R_c+R_e},\ R_h = \frac{R_7 R_e}{R_7+R_c+R_e} \quad (5)$$

同図 (6) への変形は、節点間の直列抵抗を一つにまとめるだけである。各抵抗 R_i, R_j, R_k は下記の通りになる (式 6)。

$$R_i = R_3 + R_b + R_f,\ R_j = R_4 + R_g,\ R_i = R_5 + R_h \quad (6)$$

同図 (6) の右側のデルタをスター接続に変形する (式 7)。

$$R_l = \frac{R_2 R_j}{R_2+R_j+R_k},\ R_n = \frac{R_j R_k}{R_2+R_j+R_k},\ R_m = \frac{R_2 R_k}{R_7+R_c+R_e} \quad (7)$$

同図 (7) の左側の抵抗をひとつの抵抗 R_0 にまとめると以下の通りである (式 7)。

$$R_o = \frac{(R_1+R_l)(R_i+R_n)}{R_1+R_l+R_i+R_n} \quad (7)$$

したがって、同図 (9) に示す様に AB 間を一つの抵抗にすることができる。ゆえに、AB 間の合成抵抗 R は以下の通りである (式 8)。

$$R = R_o + R_m \quad (8)$$

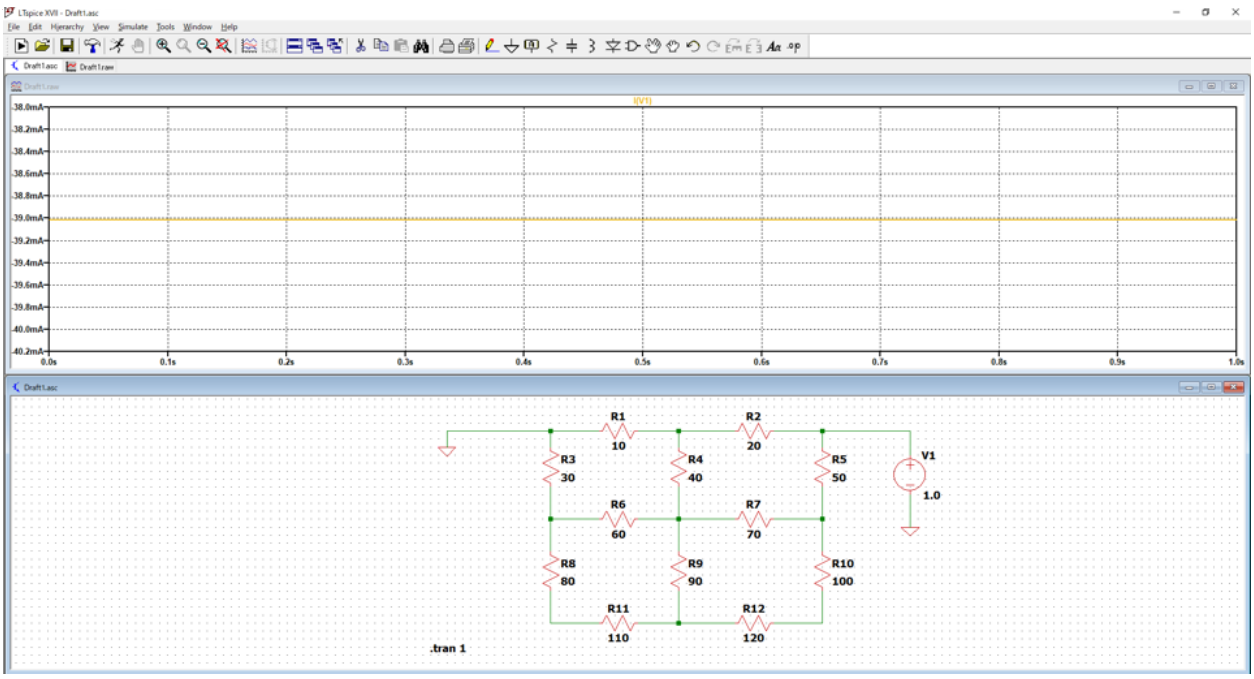


図 40 LTspice を用いた合成抵抗の計算

合成抵抗の計算が正しいか確かめるために LTspice を用いて図 40 の様な回路を考えた。ここでは、DC1V を B 節点に印加し、A 節点は接地している。電源から流れる電流値を求めることで合成抵抗を知ることができる。その結果、電流は 39 mA であった。ゆえに、LTspice で求めた抵抗値は 25.6 Ω となる。

次に、Flutter で合成抵抗の計算式を用いて計算した結果を図 41 に示している。合成抵抗は、25.6 Ω となった。したがって、合成抵抗の計算は間違っていない。

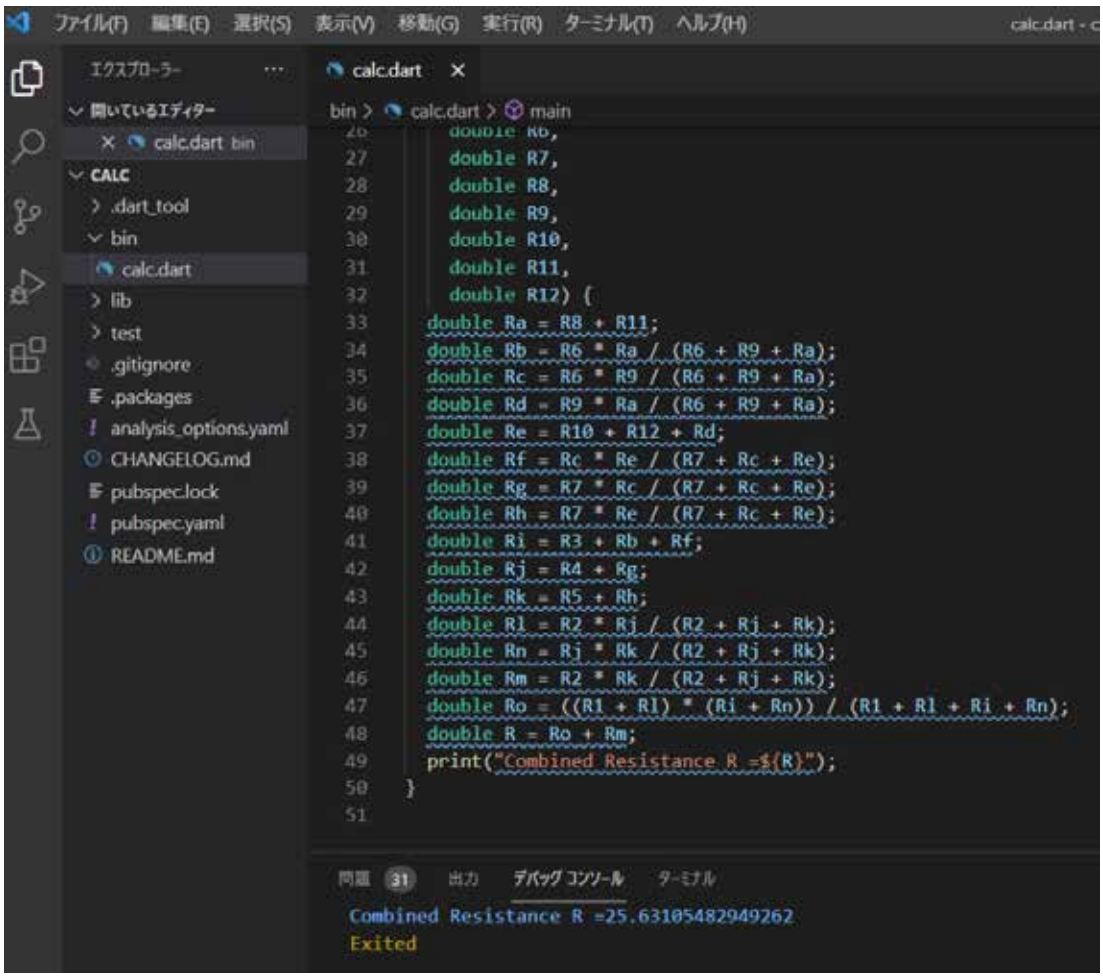


図 41 Flutter で計算した結果

```
676  // ignore: non_constant_identifier_names
677  void ReValue() {
678    do {
679      var rand = new math.Random();
680      // ignore: non_constant_identifier_names
681      R1_value = rand.nextInt(7) * 10.0;
682      R1_string = R1_value.toStringAsFixed(0);
683      if (R1_value == 60.0) R1_value = R_MAX;
684      // ignore: non_constant_identifier_names
685      R2_value = rand.nextInt(7) * 10.0;
686      R2_string = R2_value.toStringAsFixed(0);
687      if (R2_value == 60.0) R2_value = R_MAX;
688      // ignore: non_constant_identifier_names
689      R3_value = rand.nextInt(7) * 10.0;
690      R3_string = R3_value.toStringAsFixed(0);
691      if (R3_value == 60.0) R3_value = R_MAX;
692      // ignore: non_constant_identifier_names
```

図 42 抵抗値をランダムに決める処理

図 42 に示す様に、抵抗値は 0, 10, 20, 30, 40, 50Ω からランダムに選ばれる。また、抵抗が無い状態は、抵抗値として $10^{12}\Omega$ という大きな値とすることで実現している。

```
728
729     //合成抵抗の計算
730     CombResistanceCalc(
731         R1_value,
732         R2_value,
733         R3_value,
734         R4_value,
735         R5_value,
736         R6_value,
737         R7_value,
738         R8_value,
739         R9_value,
740         R10_value,
741         R11_value,
742         R12_value);
743   } while (R_value > 500); // A-B間が断線している場合は再度やり直し
744
745   bool ans_check_flag;
746   do {
747     ans_check_flag = false;
748
749     //選択肢を自動生成
750     ans_choices[0] = (R_value).round();
751
752     for (var i = 1; i < 4; i++) {
753       var rand = new math.Random();
754       ans_choices[i] = (R_value).round() -
755           rand.nextInt(21).round() +
756           rand.nextInt(18).round();
757     }
758     ans_choices.sort(); //小さい順に並べる
759     print(ans_choices);
760     //選択肢の重複、不正値を検出
761     for (var i = 0; i < 4; i++) {
762       var cnt = 0;
763       for (var j = 0; j < 4; j++) {
764         if ((i != j) && (ans_choices[i] == ans_choices[j])) {
765           ans_check_flag = true;
766           print('選択肢に重複あり、再度選択肢を生成');
767         } else if (ans_choices[j] < 0) {
768           ans_check_flag = true;
769           print('選択肢に不正値あり、再度選択肢を生成');
770         }
771       }
772     }
773   } while (ans_check_flag);
```

図 43 合成抵抗値と選択肢の生成

図 43 にランダムに選ばれた抵抗値を用いて、合成抵抗の計算を行い、4 択の数値を生成するプログラムを示す。730 行目で合成抵抗の計算を行う関数に各抵抗値を渡している。プログラムは図 44 に示す。もし計算結果が 500Ω より大きい時は節点 AB 間が繋がっていない回路が生成されていると考えて、もう一度回路生成を行う。

750 行目から回答の選択肢を自動生成している。方法は、正解の抵抗値に対して、ランダムな値を加えている。その後、選択肢を画面に表示する際に小さい順に並ぶようにソートしている。

760 行目から選択肢の重複や不正値が無いか調べている。正しくない場合は、再度選択肢の自動生成を行うようにしている。

```
776     void CombResistanceCalc(
777         double R1,
778         double R2,
779         double R3,
780         double R4,
781         double R5,
782         double R6,
783         double R7,
784         double R8,
785         double R9,
786         double R10,
787         double R11,
788         double R12) {
789       double Ra = R8 + R11;
790       double Rb = R6 * Ra / (R6 + R9 + Ra);
791       double Rc = R6 * R9 / (R6 + R9 + Ra);
792       double Rd = R9 * Ra / (R6 + R9 + Ra);
793       double Re = R10 + R12 + Rd;
794       double Rf = Rc * Re / (R7 + Rc + Re);
795       double Rg = R7 * Rc / (R7 + Rc + Re);
796       double Rh = R7 * Re / (R7 + Rc + Re);
797       double Ri = R3 + Rb + Rf;
798       double Rj = R4 + Rg;
799       double Rk = R5 + Rh;
800       double Rl = R2 * Rj / (R2 + Rj + Rk);
801       double Rn = Rj * Rk / (R2 + Rj + Rk);
802       double Rm = R2 * Rk / (R2 + Rj + Rk);
803       double Ro = ((R1 + Rl) * (Ri + Rn)) / (R1 + Rl + Ri + Rn);
804       double R = Ro + Rm;
805       R_value = R;
806       print("Combined Resistance R =${R}");
807     }
808   }
809
```

図 44 合成抵抗計算

(4) 4 択文章問題

ここでは、第二種電気工事士の筆記試験対策アプリとして重要な機能である 4 択文章問題を出題する方法について説明する。まずは、どの様な、流れで学生が 4 択文章問題を回答できるのか紹介する。その後、4 択文章問題生成のために使用している CSV 形式のデータ構造、各機能のプログラムの紹介を行う。

図 45(1) に示す様に、スタート画面の問題一覧に 4 択文章問題という項目を設けている。ユーザーは 4 択文章問題のチャレンジボタンをタップすることで、回答を開始できる。問題については、予め準備している CSV ファイルを読み込んでいる。同図 (2) に示す様に問題読込中と画面に表示されている間にデータを取込んでいる。問題読込が完了すると、同図 (3) に示す様に、4 択文章問題が表示される。今回開発した 4 択文章問題の形式は、配線図問題を除いて基本同じである。画面構成は、上段に文章問題を表示するフォームがある。制限時間内に回答させるために、回答経過時間を表示する形となっている。なお、制限時間は各問題で設定可能である。下段には、4 択の選択肢が表示される。選択肢は、毎回ランダムな

順番で表示されるようになっている。本来なら、数値などをランダムに変えて表示する方が良かったが、この時は順番のみをランダムにすることとしていた。ユーザーが制限時間内に選択肢をタップすると、自動的に正解か不正解かの判定を行う。その結果、正解の場合は、同図 (4) に示す様に、正解ですと表示が出る。また、次の問題に進めるように、選択肢の下に、次の問題を解くボタンが押せるようになる。

図 45 4 択文章問題

次に、4 択文章問題に解説機能を追加した例を紹介する。図 46(1) に、理論問題で出題される電力を計算する問題を表示している。この問題表示形式は、先ほどの例と変わりはない。ここで、文章問題の表記方法について説明する。電気工事士の問題に限らず、何かしら式を使って計算する問題の文章等で、数式を表示させたい時がある。先の図 45(3) の例では、指数の部分が「mm^2」となっており、きれいな数式ではなかった。これは通常使用する Text ウィジェットでは、指数や、一部の文字だけイタリック体にすることができない。TextSpan を使えば、一部の文字だけイタリック体にできる。しかし、今回開発していた 4 択文章問題は、CSV 形式で作成した問題データベースを読み込んで、各問題画面に表示する方法を採用している。したがって、個別対応ができないため TextSpan は使えなかった。解決策として、StyledText パッケージを使うことにした。図 47 に使用例を示している。StyledText では、表示させたい文字列の中で、イタリック体にしたい文字列を R のように囲むことで部分的に書体を変更できる。また、 のタグは、オリジナルなものも設定できて便利である。例では、文章中で、「電圧 V [V] を加えると」や「**間違っているもの**」が、それぞれ、記号がイタリック体や文字がボールド体になっていることがわかる。

(1) 4択文章問題　(2) 解説ボタン　(3) 解説表示

図 46 解説表示例

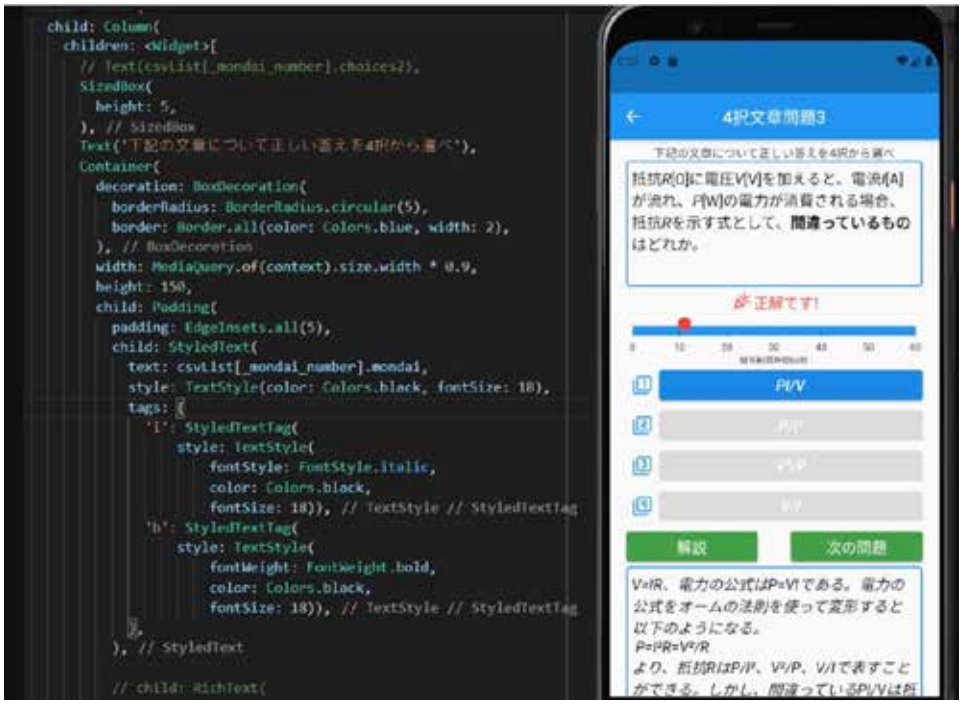


図 47 StyledText の例

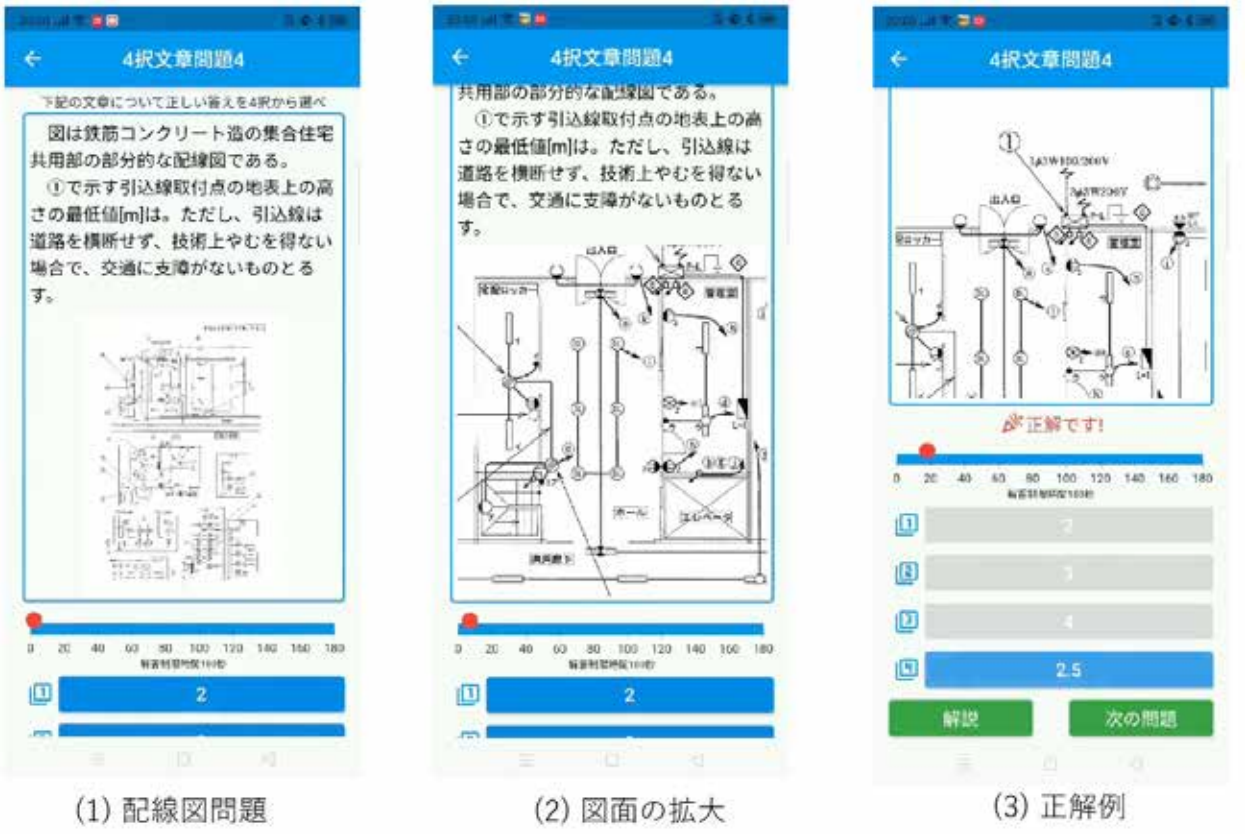


図 48 配線図問題

次に、配線図問題を紹介する。配線図問題とは、住宅の間取り図に電気機器の接続が示された配線図が描き込まれた図面を見ながら、各問を解く問題である。実際の試験では、A3 用紙サイズ 1 枚ほどの紙面に配線図が提示されている。それを切り取るなどして回答する。そのため、スマートフォンという小さい画面内で、どの様な仕組みとすると操作性を損なわず、問題を回答することができるか考えていた。作成した機能としては、図面を指で拡大縮小、移動させることができるように設計した。実際にどの様な感じになっているのか図 48 に例を示している。同図 (1) が最初に表示された状態である。配線図は全体がわかるサイズとなっている。指で図面をスワイプすると移動させることができ、ピンチ (2 本の指を画面上に置き、近づける操作がピンチイン、広げる操作がピンチアウト) 操作で直感的に図面の拡大と縮小がスムーズに行えるようになっている (同図 (2) や (3))。

1	120	直径2.6mm、長さ20mの銅線と抵抗値が最も近い同材質の銅導線はどれか。	断面積5.5mm²、長さ20m	断面積8mm²、長さ40m	断面積8mm²、長さ20m	断面積5.5mm²、長さ40m	断面積5.5mm²、長さ20m	0	抵抗値Rは以下の式より求めることができる。¥n R = ρS/L ¥n ここで、Sは断面積。Lは長さである。問題の断面積と長さを代入して計算すると良い。
2	60	消費電力が400Wの電熱器を1時間20分使用したときの発熱量[kJ]はどれか。	1920	960	1920	2400	2700	images/monda|2.jpg	発熱量Qは以下の式で求める。¥n Q = Wt¥n ここで、Wは消費電力[W]、tは時間[s]である。したがって、発熱量は¥n Q = 400 * 4800 = 1920 [kJ]となる。
3	60	抵抗R[Ω]に電圧V[V]を加えると、電流I[A]が流れ、P[W]の	PI/V	PI/V	P/I²	V²/P	V/I	0	オームの法則はV=IR、電力の公式はP=VIである。電力の公式をオームの法則を使って変形すると以下のようになる。¥n P=I²R=V²/R¥n 抵抗RはP/I²、V²/P、V/Iで表

図 49 問題データ形式

4 択文章問題のデータはあらかじめ CSV 形式でデータを作って使用している。図 49 に問題データを示す。データ構造として、各行が何番目の問題かを表している。列については、A 列から順番に、問題番号、制限時間 (s)、問題文章、正解の選択肢、選択肢 1、選択肢 2、選択肢 3、選択肢 4、画像データの保存先、解説文章となっている。この様な形式のデータを準備して、プログラムから読み込んで使用している。この方法で、とりあえず問題を準備することができるが、問題作成者として、文章中に書体を変えるためにタグを入れないといけないため不便である。この辺は、別途、バッチ処理を作り、タグ無し書いた文章から一括変換する方法などを検討している。

```
76  void csvRead() async {
77    if (_csv_read_flag == true) {
78      _csv_read_flag = false;
79      csvList = await getCsvData('assets/mondai.csv');
80      for (var i = 0; i < csvList.length; i++) {
81        print(csvList[i].id);
82        print(csvList[i].mondai);
83        print('correct Ans: ' + csvList[i].correctAnswer);
84        print('1: ' + csvList[i].choices1);
85        print('2: ' + csvList[i].choices2);
86        print('3: ' + csvList[i].choices3);
87        print('4: ' + csvList[i].choices4);
88        print('image: ' + csvList[i].mondaiImage.toString());
89        print('explanation: ' + csvList[i].explanation);
90      }
91
92      //選択肢を取り出す
93      choicesList[0] = csvList[0].choices1;
94      choicesList[1] = csvList[0].choices2;
95      choicesList[2] = csvList[0].choices3;
96      choicesList[3] = csvList[0].choices4;
97      //選択肢をシャッフル
98      choicesList.shuffle();
99
100     //制限時間
101     limitTime = csvList[_mondai_number].time;
102   }
103 }
```

図 50 問題データの読み込み

問題データの読み込み処理についてのプログラムを図 50 に示している。79 行目で準備しておいた CSV 形式の問題データを読み込む。ここでは、getCsvData という関数を自作している。この関数の仕事は、CSV 形式のデータから各問題に分割し、データをリストに入れることである。具体的なコードは図 51 に示している。

843 行目で、指定されたパスから CSV ファイルを読み込み、一つの文字列として取得する。そして、851 行目で、読み込んだ文字列データを改行文字「¥n」で切り分ける処理を行う。その後、各問題の文字列データには、カンマ「,」が含まれているので、カンマ「,」を目印に、各問題の問題番号や制限時間などのデータを切り分けて、データ保存用の配列に入れる処理を行っている。この処理が終わると、getCsvData の戻り値としてデータ配列が図 49 の 79 行目の csvList に代入される。

次に、図 49 の 80 行から 102 行目で、文章問題の選択肢を表示するために使用している配列にデータを取り出している。ここで、選択肢の順番がランダムになるように shuffle メソッドを使っている。最後に、制限時間を取り出している。これで、表示させたい問題のデータを準備することができた。なお、画像データは、別のプログラムで読み込ませている。その他のプログラムは、長くなるので掲載は割愛する。単純に、問題データを表示させているだけである。

```dart
Future<List<CsvData>> getCsvData(String path) async {
  // 戻り値を生成
  List<CsvData> list = [];

  // csvデータを全て読み込む
  String csv = await rootBundle.loadString(path);

  print('csv file');
  print(csv);

  print('\nrows');

  // csvデータを1行ずつ処理する
  for (String line in csv.split("\n")) {
    // カンマ区切りで各列のデータを配列に格納
    List rows = line.split(','); // split by comma
    print(rows);

    // csvデータを生成
    CsvData rowData = CsvData(
        id: int.parse(rows[0]),
        time: int.parse(rows[1]),
        mondai: rows[2],
        correctAnswer: rows[3],
        choices1: rows[4],
        choices2: rows[5],
        choices3: rows[6],
        choices4: rows[7],
        mondaiImage: rows[8],
        explanation: rows[9]);

    // csvデータをリストに格納
    list.add(rowData);
  }

  print('\nlist');
  print(list);

  // リターン
  return list;
}
```

図 51 データの取出し

(5) チャレンジログ機能 (学習の振り返り)

ここでは、チャレンジログ機能と呼んでいる学生が様々問題を回答したログを見て学習の振り返りが行える仕組みについて説明する。図 52(1) に、スタート画面を示している。スタート画面の下部にチャレンジログ機能を使うためのアイコンを設置している。アイコンをタップすると、同図 (2) に遷移する。この画面では、各問題カテゴリのログが見れるようになっている。また、カテゴリ別に過去の回答結果を基に正解率を表示している。ここでは、4 択文章問題のログを見てみる。同図 (3) に 4 択文章問題を過去に解いた結果一覧を示している。このログでは、問題を回答した日時及び正解か不正解だったかを文字とアイコンで表示されるようにしている。

(1) チャレンジログボタン　(2) ログ表示　(3) 4択文章問題回答のログ

図 52 チャレンジログ画面

(1) 合成抵抗の計算ログ　(2) 写真ログ　(3) 配線図問題回答のログ

図 53 合成抵抗問題や配線図問題のログ表示

次に、図 52(3) の項目をタップすると、図 53(1) の様に、過去に回答した計算問題の解答が見られる。手書きで描いていた図や回答した文章がそのまま表示される。さらに、カメラで撮影したデータもスクロールすることで確認できる (同図 (2))。配線図問題の回答結果についても、同様に見ることができる (同図 (3))。なお、これらのログデータは、スマートフォン内に保存されている。

(6) 写真鑑別問題

第二種電気工事士の筆記試験では、器具等を見て、名称や用途が問われる写真鑑別問題が出題される。この問題は、知っているだけで回答できるが、数が多いので暗記するのが大変である。そこで、暗記するのをサポートする機能として、写真鑑別問題を作成している。この写真鑑別問題のプログラムは、2022 年度 貞方研究室の西田君が作成したものである。

図 54 写真鑑別問題の概要

図 55 データファイル構造

写真鑑別問題の流れを図 54 に示している。スタート画面のスタートボタンを押すと、第 1 問目が表示される。全部で 100 問ほどあり、全て回答を終えると、結果画面が表示される。この画面には、正解率と学習者への励ましの言葉が表示されるようになっている。

次に、問題のデータ構造について説明する。図 55 に CSV データなどを示している。データフォーマットとしては、先に説明したものとあまり変わりはない。各行で、問題暗号、正解番号、選択肢 1 ～ 4、画像データのパスとなっている。CSV データの読み込み方法等は、先に説明した手順で行っている。

最後に、結果を表示する部分のプログラムを紹介する。図 56 に示す様に、正解数は result 変数、問題数は quizNumber 変数に入っている。それらを使って計算した結果を表示している。メッセージ内容は、正解率に応じて 5 段階で内容が変わるように条件分岐させている (図 57)。

図 56 結果の表示

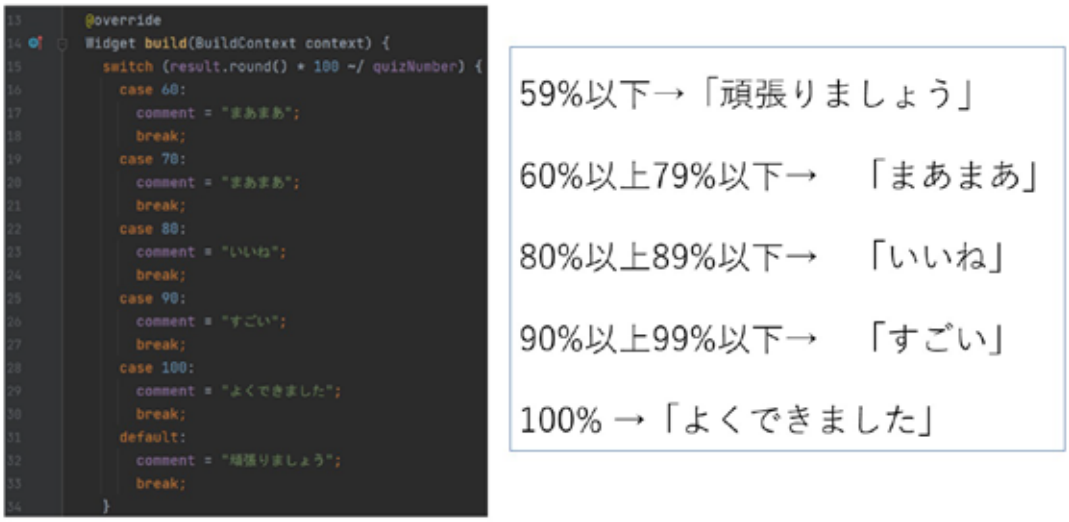


図 57 メッセージの切り替え

(7) 資料確認機能

スタート画面の下部に設置している講義アイコンを押すことで、講義で使用した資料や講義動画を見ることができる。講義資料は PDF で開くことができる。また、講義動画は、Microsoft Stream にアップしている動画の URL をブラウザで開くようにしている (図 58)。

図 58 資料確認画面

(8) 質問機能

図 59 に質問受付画面を示している。スタート画面から質問アイコンをタップすると、Microsoft Forms の質問画面がブラウザで開く。あとは、氏名等を入力し送信すると担当者に連絡が届く仕組みである。

図 59 質問機能

7 まとめ

本研究開発報告では、第二種電気工事士の筆記試験対策アプリの開発と題して取り組んだ内容を紹介した。本研究開発のきっかけは、電気工事実習という実習科目で資格を取得して電気工事関連の企業へ就職したい学生を応援するために、何か良い方法はないだろうかと考えたことから始まっている。第 2 章では、多くのページを使用して、工学部電気情報工学科で取り組まれてきた学生のキャリア支援としての電気工事士資格取得支援活動の歴史、そして、理工学部電気工学科となり電気工事実習という電気工事士の資格取得支援に特化した実習科目設置の背景をお伝えすることができたのではないかと思っている。

本題の第二種電気工事士の筆記試験対策アプリ開発については、アプリの仕様を決めて開発を行っていたが、最終的には完成させることはできていない。基本的な機能については構築ができたと考えているが、アプリとしての形にはなっていない。その原因は、私の勉強不足が主な所である。折角、研究開発のチャンスを頂いたにもかかわらず完成できず残念である。しかしながら、今回得た知識等は、別の形で活かせると考えている。そもそも、電子物性に楽しさを覚え、有機半導体関連を研究している私が、何故、アプリ開発に取り組んでいるか？その答えは、将来、皮膚に貼付けて使用するペースタブルヘルスケアデバイスの開発を目指しているからである。この研究は、デバイスだけで完成しても意味が無い。ユーザーが測定データについてスマホアプリでグラフを確認し健康維持ができることが重要となる。そのため、幅広い知識や技術を身に付ける必要がある。今回の研究開発は、その一環としても取組ませて頂いていた。

8 今後の課題と展開

今後の課題としては、筆記試験対策アプリを完成させることであるが、今回の開発では、技能試験対策については考えていなかった。技能試験では、公表問題 13 課題の中から 1 課題が本番で出題される。制限時間 40 分で、欠陥なく完成させる必要がある。技能を習得するのが合格への近道であるが、ちょっとしたことが欠陥となり、不合格になる学生も多い。

図 60 に示す写真には、複数の電線の銅線をリングスリーブという金具を圧着工具で圧着することで電気的につなげる作業の注意点をまとめている。一見すると単純な作業に見えるが、銅線の直径や本数の組み合わせでリングスリーブのサイズや側面の刻印が異なる。また、リングスリーブで電線の被覆を噛んではいけないなどの決まりがある。また、図 61 に示すのは電球を接続して使うランプレセプタクルという器具である。ランプレセプタクルは基本的な器具であり、どの課題にも出題される。この器具へ電線の銅線を 2 本取付する。その時の決まりごとが数個あるためミスを犯しやすい。そのため、技能試験対策では、ミスを無くすために、先生方が入念に確認を行い、間違いが有れば学生に指導している。この指導は、人手が必要で

大変である。特に、学生が一人で練習している時は、だれも確認することができないこともある。そこで、スマートフォンのカメラで各部を撮影することで欠陥を見つけることができれば学生だけでも作業の確認が行える。機械学習などを使って､実現できないかと考えている。

図 60 リングスリーブを用いた電線の圧着

図 61 ランプレセプタクルへの電線の接続

9 謝辞

Flutter 開発については、共同開発者として参加して頂いていた山本氏に感謝申し上げます。忙しい中、アドバイスを頂き進めることができました。私の力不足でリリースまで行けなかったことが残念ですが、次に何か共同でできることがあればよろしくお願いいたします。

第一種及び第二種電気工事士の資格取得支援活動は、九州産業大学が売りにしている Advanced Program に認定されるほどの活動になっている。これは、理工学部電気工学科の技能員の先生をはじめ、学部学科教職員のご理解とご支援、学生アルバイタ、大学からのバックアップなしには成すことはできない。この場を借りて感謝申し上げます。今後とも、様々な面で学生支援に力を入れていくと考えています。

最後に、本研究開発のチャンスを与えて頂いた九州産業大学　総合情報基盤センターのご担当者様へには、報告書の提出が遅れに遅れたこと深くお詫び申し上げます。今後ともよろしくお願いいたします。

引用・参考文献

[1] 電気工事士試験に関する Web サイト ,『一般財団法人　電気技術者試験センター』,
URL: https://www.shiken.or.jp/index.html (最終観覧日 :2023 年 11 月 1 日).

[2] 東京消防庁 ,『電気火災を防ごう』,
URL: https://www.tfd.metro.tokyo.lg.jp/camp/2022/202208/camp3.html
(最終観覧日 :2023 年 11 月 1 日).

[3] 労働安全衛生総合研究所 ,『感電の基礎と過去 30 年間の死亡災害の統計』,
URL: https://www.jniosh.johas.go.jp/publication/doc/td/SD-No25.pdf
(取得日 :2023 年 11 月 1 日).

[4] 経済産業省　産業保安グループ　電力安全課 ,
『電気保安人材の中長期的な確保に 向けた課題と対応の方向性について』,
URL: https://www.meti.go.jp/shingikai/sankoshin/hoan_shohi/denryoku_anzen/pdf/016_05_00.pdf
(取得日 :2023 年 11 月 1 日).

[5] 国土交通省 ,『スマートシティ官民連携プラットフォーム』,
URL: https://www.mlit.go.jp/scpf/
(最終観覧日 :2023 年 11 月 1 日).

[6] 電気技術者センター ,『第二種電気工事士試験』,
URL: https://www.shiken.or.jp/examination/index05.html
(最終観覧日 :2023 年 11 月 1 日).

[7] 電気技術者センター ,『第一種電気工事士試験』,
URL: https://www.shiken.or.jp/examination/index04.html
(最終観覧日 :2023 年 11 月 1 日).

[8] オーム社編 ,『2023 年度版　第二種電気工事士　筆記試験　標準解答集』,
オーム社 , 2022.

[9] 経済産業省令 電気工事法 ,『学科試験の免除』,
URL: https://elaws.e-gov.go.jp/document?lawid=335CO0000000260
(最終観覧日 :2023 年 11 月 1 日).

[10] 電気技術者センター ,『2023 年度第二種電気工事士筆記試験下期午前』,
URL: https://www.shiken.or.jp/answer/pdf/383/file_nm01/2023am_K_shimokigakka.pdf
(最終観覧日 :2023 年 11 月 1 日).

[11] 電気技術者センター ,『第二種電気工事士の技能試験の公表問題』,
URL: https://www.shiken.or.jp/ginouanswerK/ginou20230723.html
(最終観覧日 :2023 年 11 月 1 日).

[12] 電気技術者センター ,『第二種電気工事士受験者数及び合格者数の推移』,
URL: https://www.shiken.or.jp/situation/s-construction02.html
(最終観覧日 :2023 年 11 月 1 日).
[13] 九州産業大学 ,『Advanced Program』,
URL: https://www.kyusan-u.ac.jp/nyushi/adovancedprogram/
(最終観覧日 :2023 年 11 月 1 日).
[14] Flutter,『Flutter』, URL: https://flutter.dev/
(最終観覧日 :2023 年 11 月 1 日).
[15] 石井　幸次 ,『基礎から学ぶ Flutter』,
C&R 研究所 , 2021.
[16] 新井　克人 ,『はじめての Flutter』,
工学社 , 2020.
[17] 澤　良弘 , 上村　隆弘 , 村岡　直人 , 多田　幸一 ,
『Flutter 開発入門』, マイナビ出版 , 2021.