Keytect：低負荷なインタラクションによる鍵の自動管理システム

三井田幸貴[†1]　畠山彩音[†1]　益子元太[†1]　緑川育人[†1]　安細勉[†1]　奥出真理子[†1]
周而晶[†1]　丸山智章[†1]

概要：現在，多くの学校では，学校の鍵の管理者が各種施設の鍵をアナログ的に管理しているが，これはセキュリティ面において不十分であり，利用者の負荷も大きい．本研究では，問題解決のために，鍵の貸し出し，返却の状況をいかなる場所においても把握できるシステムとユーザ認証を行なわない限り物理的に鍵を取り出すことが不可能となるロック機構付きのケースの作成を行なった．結果として，本システムは学校が管理している鍵の不正利用防止に十分実用的であり，また利用者の負荷も削減することができたと結論付けられる．

1. はじめに

1.1 研究の背景

学校には，教室や体育館，音楽室，武道場など様々な施設があり，授業や課外活動などに利用されている．多くの学校においては，それら施設が使われていないときには施錠されており，学生が利用する際には学校から鍵を借りて利用している．例えばある学校では，部活動等で施設の鍵を利用する際に，事務室等の鍵の管理担当者のいる場所に設置された紙媒体の鍵貸し出し記録表に学年や現在時刻，氏名を記入して各自で鍵を持ち出すシステムとなっている．借用した鍵を返却する際も同様に記録表に記入し，所定の位置に鍵を戻すという形をとっている．

しかし，そのシステムにはいくつか問題点が存在する．具体的には，同校の学生より次の 2 点の問題点が挙げられた：(1) 目的の鍵が既に貸し出されているかどうか足を運ばなければ確認できない，(2) 借用した鍵を誤って校外へ持ち出す恐れがある．また，毎日様々な教員や学生による鍵の貸し出し状況を記録するにあたり，多くの紙が使用されることは避けられない．環境問題やスマートなデータ管理の観点からペーパーレス化が推進される昨今，紙媒体ではなく電子媒体での管理に切り替えることが望ましい．

1.2 研究の目的

本研究では，前述の問題点を改善するために鍵の自動管理システム「Keytect」を提案する．これにより，鍵貸し出し業務を情報処理技術により支援し，問題の緩和及び解決を目指す．具体的には，少ない手順で記録表への記入を自動化し，ユーザに対して低い負荷で鍵貸し出し及び返却の手続きが済むようにすることを目指す．また，Web アプリケーション等の電子媒体により貸し出し状態を把握できるようにし，問題点 (1) の解決を目指す．さらに，鍵貸し出し履歴を処理することにより，誤って鍵を持ち帰ってしまった等の異常状態を自律的に検知し，鍵の管理者や当事者である学生に通知することで，問題点 (2) による悪影響を緩和することを目指す．

2. システム概要

2.1 全体の概要

まず，ユーザは Keytect に自身の ID と自身に対応するパスワード（以下，「認証情報」と表記）を入力する．Keytect は，この情報の組み合わせが自身のもつユーザ一覧に存在するかを確認し，存在すれば認証成功としてユーザに鍵の取り出しを促す．存在しなければ，認証失敗としてもう一度ユーザに認証情報の入力を促す．

2.2 ロック機構

Keytect では，誰が鍵を借りたかを確実に把握するため，鍵はロック機構付きのケース内に入れられている．初期状態では，この鍵は施錠されていて物理的に鍵が取り出せない状態になっている（図 1）．ロック解除のためには，併設されている画面の指示に従って認証情報を入力する必要がある．入力された認証情報が Keytect 内に保存されているユーザ一覧内に存在した場合は，ロックが解除され，ユーザがドアを開けられるようになり，鍵が利用できるようになる（図 2）．図 2 の状態では誰が Keytect を操作しているかがわかっているため，この状態の間に起こった鍵の取り出し状態の変化と認証情報を組み合わせて，誰がどの鍵を持ち出したか，あるいは返したかを把握する．

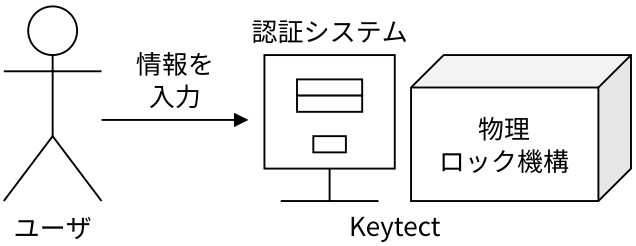

図 1　ロック状態の Keytect とユーザの操作

†1 茨城工業高等専門学校，国際創造工学科，情報系

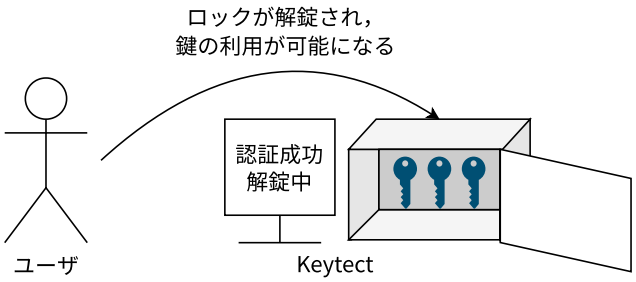

図 2　ロック解除状態の Keytect とユーザの操作

3. 課題

本研究を進めていくにあたり，以下の 3 項目の課題が浮上した：(1) 鍵を借りる人と使う人が異なる場合の措置，(2) 災害時のフェイルセーフ，(3) 入力ミスによる記録の異常．

3.1 鍵を借りる人と使う人が異なる場合の措置

「借りる人と使う人が異なる場合」とは，貸し出しの委託等による鍵の横流しが発生した場合を指す．その事象の問題点として，鍵を借りた人が鍵の所在を把握していない可能性が存在することが挙げられる．

この課題に対するアプローチとして，鍵の貸し出し状況を表示するページに「借りた人」「返した人」の 2 種類を掲載することによる鍵の流れの明確化を図る．また，「借りた人」と「返した人」が一致しなかった場合，「借りた人」に対してその事実をメール等で通知する．鍵が長時間返却されなかった場合も，「借りた人」に対しての通知が行われる．このようにして，鍵を借りた人と実際に使った人の間の連絡を促す．

3.2 停電時のフェイルセーフ

前述したドアロックはサーボモータ等を利用して実装を行ったため，停電などの異常事態時はロックが解除できなくなる．これにより完全に鍵が利用できなくなると，ドアロック機構あるいはケースそのものを破壊する以外に鍵を取り出す方法がなくなってしまう．

これを防ぐため，ユーザが通常使用するドアをロックする機構とは別に，背面に特別なマスターキーでのみロック解除できる，フェイルセーフ用のロック機構を搭載することにした．このマスターキーは，学校の鍵の管理者によって厳重に管理されることを想定している．

3.3 入力ミスによる記録の異常

実装に着手した当初は鍵の貸し出し，返却を行う際に記入するフォームを作成し，「貸し出し」「返却」をラジオボタンで選択する方式を採っていたが，「貸し出しのはずが返却を選択してしまった」，もしくは逆の事例が発生する可能性が問題点として挙げられた．

実際にこの問題が発生すると，既に返却された鍵が再び返却された，といった想定していない記録がなされてしまうこととなる．

この問題の解決方法として，データベースで情報を参照し，意図しない入力がなされた際はエラー画面を表示させる案も浮上したが，実装の容易さや利用者の快適さの観点から，「貸し出し用入力フォーム」と「返却用入力フォーム」の 2 種類に分けてページを作成することで，そもそもの入力ミスの防止を試みる案が採用された．

4. 実装

まず，4.2 節までで，システム全体に及ぶ実装上の議論とその結論を述べる．その後，4.3 節から 4.6 節で鍵の検知やドアロック等のハードウェアに関わる実装上の議論と結論を述べ，4.7 節から 4.11 節でソフトウェアに関わる議論と結論を述べる．

4.1 動作のフロー

Keytect の動作フローを図 3 に示す．

4.2 ハードウェアと基本的な技術スタック

Keytect は，Raspberry Pi 4 上でプログラムを稼働させ，その GPIO を用いて電気回路を制御しユーザとインタラクションを行うこととした．実際に制御を行うソフトウェアの実装に用いる言語は， Raspberry Pi 4 の GPIO との連携用のライブラリである「RPi.GPIO」が提供されており [1]，ハードウェアとの連携が容易に実装できることが期待できたため Python 3 を採用した．

4.3 鍵の検知の手法

鍵の検知の実装方針を決定する際，3 つの手法が挙げられた：（1）重さセンサを用いて検知を行う方法，(2) フォトインタラプタ（透過型フォトセンサ）を用いて検知を行う方法，(3) フォトリフレクタ（反射型フォトセンサ）を用いて検知を行う方法．

（1）重さセンサを用いた検知については，鍵の個体による重さの差や，そもそも鍵が軽すぎて検知できないといったことが懸念された．それに対し，(2) 及び (3) のように光センサを用いる方法は，鍵が光の反射や遮光に関与しているか否かという，よりデジタルな情報を得ることができ，今回のケースと相性がより良いと判断したため，重さセンサを用いた検知の手法は棄却された．

(2) フォトインタラプタを用いた検知については，EE-SX484 を用いて検証を行った．参考サイト [2]によると，フォトインタラプタは，「光を出す発光素子の発光面と光を受けて信号を出す受光素子の受光面が対向しており、その間の空間を物体が通過して、光をさえぎることにより受光素子の受光量が変わるようになっています。」とある．これによれば発光面と受光部の間で光を遮光しているかどうかという，他の方法と比較してよりデジタルな情報から物の有無を判定することができるため，他の方法よりも安定性が高いと推測した．一方で，フォトインタラプタは，その原理や形状の都合上鍵を元の場所に戻しにくく，現在

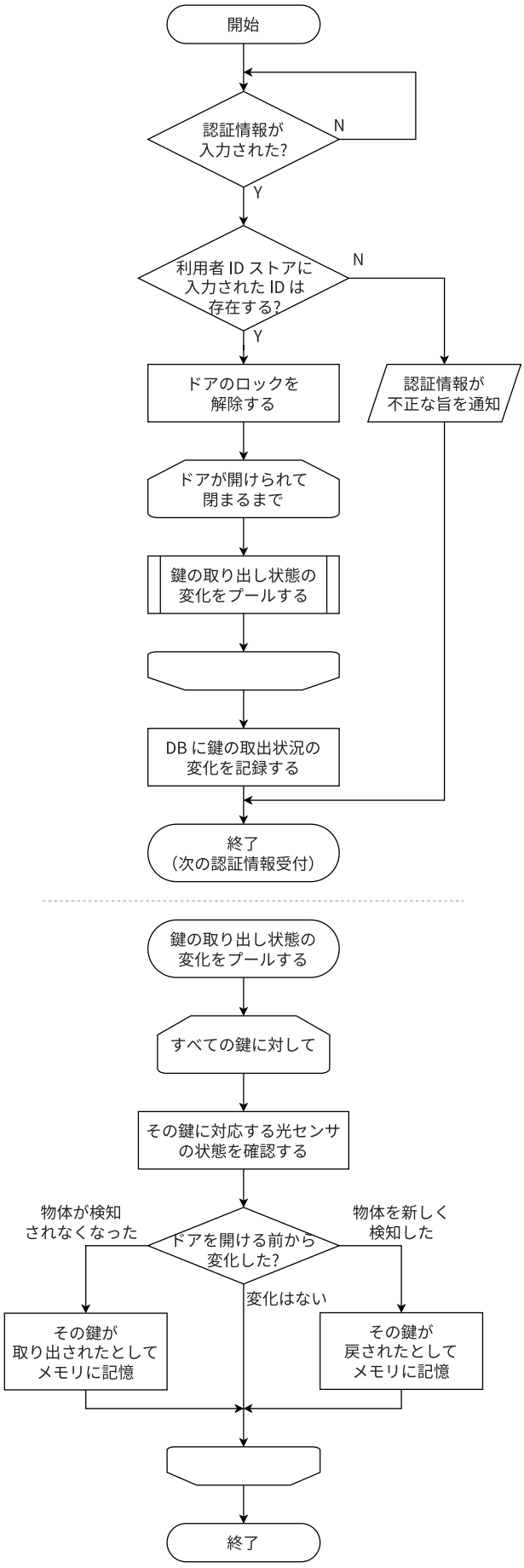

図 3　Keytect の動作フロー図

の「鍵をフックに掛ける」という利用者のフローをそのまま Keytect でも再現しようとすると利用時の煩わしさを生んでしまう．

（3）フォトリフレクタを用いた検知については，TPR-105F を用いて検証を行った．参考サイト [2]によれば,「発光素子の発光面と受光素子の受光面が同一方向になるように取りつけられています。そして、検出物体が光路を通過すると、発光面を出た光が検出物体にあたり、そこからの反射光が受光素子に照射され、受光素子の受光量が変化するようになっています。」とある．これによれば，フォトインタラプタのような形状に起因する利用時の煩わしさの問題は起こらず,「鍵をフックに掛ける」という利用者のフローにうまく適合できる．

一方で，フォトリフレクタはフォトインタラプタに比べ,発光部以外から発せされた光に反応したり，あるいは鍵ではなく壁に光が反射したりして誤反応する，といったリスクが存在する [2]. それでも，このような問題点はケースの形状等を工夫することである程度回避することができる．そのため，精度は今回の用途において十分であり，形状としてもユーザの不便を生むものでもないと判断し，（3）を採択した．

4.4 複数の鍵の検知

一般的に管理対象となる鍵は施設の数だけ存在し，その数は数十個となる．単純に各センサを GPIO ピンに接続するとすぐにピンが枯渇してしまう．

この問題を解決するため，アナログ信号のマルチプレクサを用いた．今回は MCP3002 を用いることとした．MCP3002 は，2 つのアナログ信号の受け口（チャンネル）を設けており，SPI プロトコルを介してどのチャンネルに接続されている信号を取得するかを選択することができる [3]．これにより，1 つの IC で 2 つのフォトリフレクタを取り扱うことができるようになる．

実際に鍵が数十個あることを考えると 2 つのチャンネルのみでは足りない一方，類似の IC として 8 つのチャンネルを扱える MCP3008 が存在する [4]他，プロトコルが異なるが CD74HC4067 等の更に多量なアナログ信号を扱えるマルチプレクサも存在する [5]ため，実用に移す際は同等の設計で大きな問題は起こらないと判断し，今回は MCP3002 で 2 つの鍵を取り扱うこととした．

MCP3002 を用いて複数のフォトリフレクタを取り扱う際の回路図を図 4 に示す [3] [6]．ここで，Raspberry Pi 4 のピンは MCP3002 と接続されているもののみを抜粋して表示している．

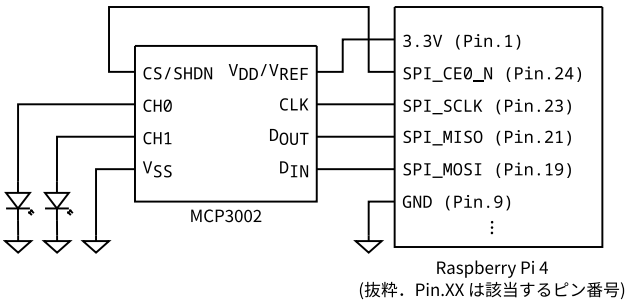

図 4　MCP3002 を用いて複数のフォトリフレクタを取り扱う際の回路図

4.5 ドアロックの実装

ドアにロックを実装するにあたり，サーボモータで制御できるロックを搭載する必要があった．今回は，ドアにサーボモータを取り付け，さらにサーボモータに細い棒を取り付け，壁に細長い穴を開けてそこに棒を通すことで扉が開かないようにした．また，十分に穴を細長くすることで，サーボモータの回転によって細い棒を穴から引き抜くことができるようにし，ロックを解錠できるようにした．

サーボモータが稼働した際にロックが解錠される様子を図 5 に示す．赤色の直方体が壁，オレンジ色の円柱がサーボモータ，青色の細い立方体がサーボモータに取り付けられた細長い棒である．ここで，「ドア部」と「側面」と床以外の壁（天井など）は省略している．

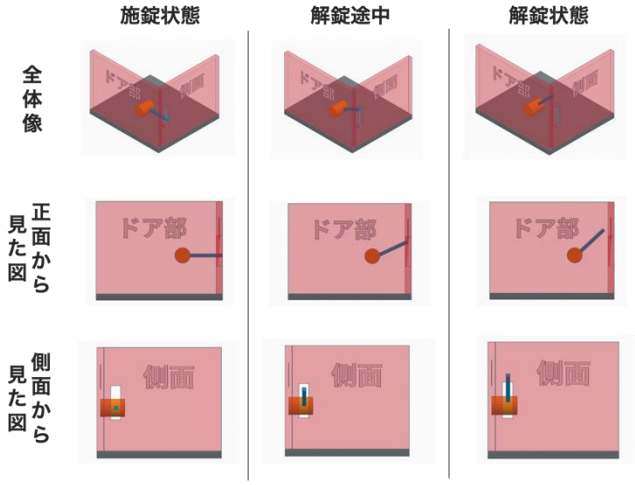

図 5　サーボモータが稼働した際の解錠の様子

4.6 バックアップ用のロック機構

3.2 節で議論したように，災害などで電気が利用できず，正面のロックが動かなくなってしまった場合に備え，別の方法でロックを解放できる手段が必要である．

これに対応するため，ケースの上面を開けられるようにし，側面と南京錠で固定することにした．南京錠の鍵は，鍵の管理者によって厳重に管理されることを想定している．

4.7 ソフトウェアの設計の概覧

Keytect は複数人で並行して開発した．複数人で並行して開発するにあたり，一つのモノリシックなモジュールを全員が開発するよりも，モジュールを分割して各人が開発するモジュールを分散させるほうが望ましい．

特に，GPIO やデータベースの外部リソースは，利用に先立って初期化が必要（たとえば，データベースであれば接続先のデータベースの URL を指定してセッションを開くなど）である．この処理が分散してしまうと思わず複数回初期化が発生するなどして，アプリケーション全体の状態が不正になるリスクが存在する．そのため，外部リソースにアクセスするロジックは特にロジックの分散や重複を排除するべきである．

このため，各サービスの開発に先んじて「インフラ層抽象化レイヤー」を実装することで未然にロジックの分散や重複を防いだ．このレイヤーからは，外部リソースにアクセスするための関数群を提供しており，各サービスはここを介して外部リソースへのアクセスを実施することができる．このようにして，外部リソースを利用するさいの依存先のモジュールを一点に集中させることで，先に述べたような状態不正を未然に防止することができる．

最終的に用いたアーキテクチャの概覧を図 6 に示す．

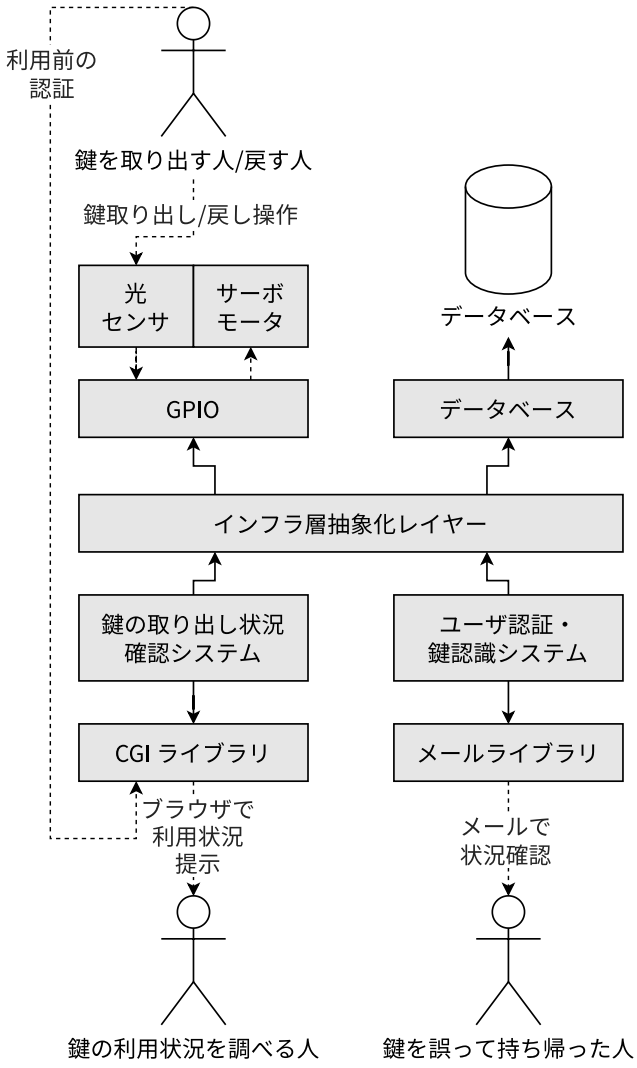

図 6　アーキテクチャの概覧

4.8 ユーザ認証

2.2 節で述べたロック解除前の認証については，備え付けのコンピュータに対し，自身の ID を入力することで，鍵を取り出そうとしているあるいは戻そうとしているユーザを認証できるようにした． ユーザ ID については，学校より学生や教員に付与される E メールアドレスのユーザ名部分（アットマークよりも前の部分）を用いることとし，新しい ID の記憶を不要とし負荷削減を図った．

この画面の開発及びホスティングは，Python 3 の CGI モジュールを用いることで行った．開発中の認証画面のスクリーンショットを図 7 に示す．

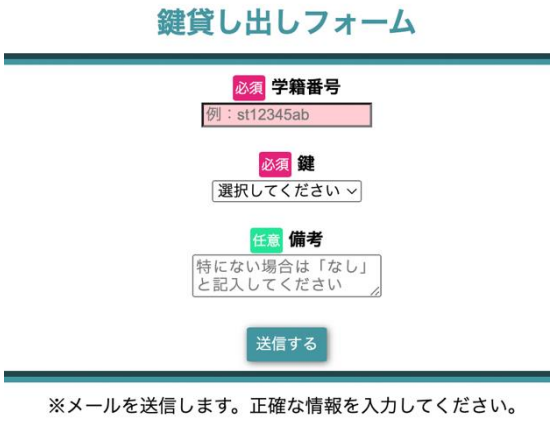

図 7 開発中の認証画面

4.9 鍵の取り出しとユーザの関連付け

図 3 に示したフローチャートの通り，ドアが開いている間は誰が認証されたかの情報が有効である．これを用いて，認証された情報と鍵のセンサの変化を対応付けて保存することで鍵の取り出しとユーザの関連付けを保存している．

鍵を一度取り出した後すぐに元に戻したなどのケースに対応するために，一度メモリ上に鍵の変化状態を記録しておき，ドアが閉まったあとにその情報を実際にデータベース上に反映させるフローを取っている．

4.10 リモートからの鍵利用状況の確認

Raspberry Pi 4 上で，利用状況確認用ページを学内に公開し，学内にいればどこでも鍵が利用されているかを確認できるようになった．利用状況確認用ページは認証画面と同様に Python 3 の CGI を用いて開発され，同じインフラ層抽象化レイヤーを介してデータベースにアクセスする．開発中のこの画面のスクリーンショットを図 8 に示す．

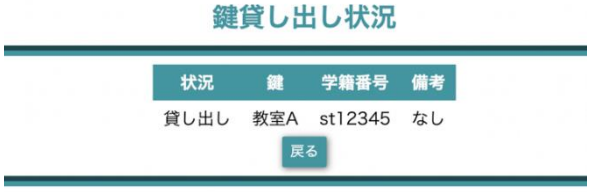

図 8 鍵の利用状況確認用ページのスクリーンショット

4.11 メールでの通知

データベースに保存された鍵の状態を定期的に解析することで，日付が変わっても鍵が返却されないといった運用上の問題が推測されたときに，速やかに当事者に連絡するためのメールシステムを実装した．

まず，3.1 節で論じた，鍵を借りた人と返した人が異なる場合，このユーザの間での情報共有を支援するために，鍵が返されドアが閉められた直後に両者に対してメールで通知を行う．また，鍵を借りた人が誤って鍵を持ち帰ってしまったり，気づかないうちに紛失していたりするなどの状態を，Keytect から能動的にユーザに通知するため，毎日終業時間後に鍵の状態を確認し，未だ返却されていない鍵が存在した場合はその鍵を借りているユーザにメール通知を送信する，さらに，揺れや鍵を戻す際の不良により鍵が落ちてしまった状態を検知し管理者に通知するため，一定時間ごとにデータベースに保存されている鍵の貸出状況とセンサの状態を比較し，2 つの間に相違があった際に管理者にメール通知を行う．

メールアドレスについては，4.8 節で述べたユーザ ID の仕組みより，ユーザ ID からメールアドレスを導出することができるため，そのユーザ ID を基に送付先メールアドレスを導出している．

5. 結果

まず，5.4 節までで，3 節に記したような開発段階で浮上した課題の解決に，4 節にて取り組んだ結果を述べる．次に，5.5 節にて，今回の研究全体を通しての結果を述べる．

5.1 鍵を借りる人と使う人が異なる場合の措置

3.1 節に記した提案のように，鍵を「借りた人」と「返した人」を比較して，一致しなかった場合，もしくは規定時間以内に鍵の返却がなされなかった場合に「借りた人」に対してその旨をメールで通知することにより，鍵の所在が明らかでない状態にある時間をほぼ一定にすることに成功した．「借りた人」に連絡を受けた「返した人」が鍵を紛失している可能性も考慮すべきであるが，その場合であっても発覚するまでに要する時間が短ければ，その分早く捜索に取り掛かることが可能である．ただし，メールを受け取った「借りた人」が「返した人」に対して連絡を行なわないなどの人的要因である望ましくない事象に対しての対策方法については，鍵の位置情報を把握できるようにするなど更に研究を重ねていく必要があるだろう．

5.2 停電時のフェイルセーフ

3.2 節に記した提案のように，「フェイルセーフ用のロック機構」を搭載したケースを作成した．それにより，災害時や予期せぬ事態でケースのドアをロックする機構の動作が停止してしまった場合であっても，破壊以外の物理的方法でロックの解除を行なうことが可能となった．

ケースを解錠するマスターキーは，実際に当校にて運用

する際は，学生課で管理して頂くことを想定しているが，マスターキーの紛失，破損の対策については現時点では考えられていないため，管理者に委ねることとなっている．

5.3 入力ミスによる記録の異常

3.3 節に記した提案のように，「貸し出し」と「返却」で情報を入力する画面を分割することで，視覚的に分かりやすくなり入力ミスの要因を減らすことに成功した．

しかし，故意に連続して同じ入力がなされる可能性も考えられるため，それが検知された際にエラー画面を表示させて送信をブロックすることでデータベースに異常が生じるのを防止した．

5.4 Keytect を利用する際のユーザの負荷と影響

Keytect は, 導入にあたって鍵を利用するユーザの負担が変化しない，あるいは減少させることを念頭において設計された．

ユーザが学校から通知されるメールアドレスより得られる ID を用いて認証のための記憶負荷を抑えたり，システムから能動的にユーザに必要な通知を送信したりすることで鍵を常にトラッキングし正常な状態に保つサポートを行った．鍵を利用する際の負荷は変化しない一方，鍵貸し出し記録への記録の自動化や，万が一の際のサポートを能動的に行うことで，鍵の利用や管理にあたっての負荷を減らすことができたと考えている．

一方で, この ID は個人の中で秘匿されるものではなく，この ID のみで認証を行うことはセキュリティ上の大きな懸念である．そのため，今後負荷上昇を抑えながらセキュリティを向上させる方法を模索していく必要がある．

5.5 全体を通しての結果

機能面に大きな問題はなく，課題の解決にも成功したことから，十分実用的であると言える結果となった．

ただし，今回作成したものはテスト用の小規模なものであり，当校の所有する鍵全てを保護することは不可能であるため，実際に運用するとなった場合は新たにケースを作成する必要がある．

また，上記の理由により，十分な量のテストデータが取れておらず，推測による部分も存在する．早急なデータ取得のための試験的運用が必要であるだろう．

最後に, Keytect は利用者の善性に委ねる部分が決して少なくない．例えば，違う場所に違う鍵を戻された場合の措置については有効な防止策が考えられていない. そのため，誰が利用する場合においても必ず正しく鍵が返却されるシステムの完成のためには更なる研究が求められる．

参考文献

[1] “RPI.GPIO - PyPI”. https://pypi.org/project/RPi.GPIO/, (最終アクセス 2022-12-17)
[2] オムロン株式会社,“フォト・マイクロセンサの技術解説”. https://components.omron.com/jp-ja/ds_related_pdf/CEWP-120.pdf, (最終アクセス 2022-12-17)
[3] Microchip Technology Inc.,“2.7V Dual Channel 10-Bit A/D Converter with SPI Serial Interface Data Sheet”. https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21294E.pdf, (最終アクセス 2022-12-17)
[4] Microchip Technology Inc.,“MCP3008 | Microchip Technology”. https://www.microchip.com/en-us/product/MCP3008, (最終アクセス 2022-12-17)
[5] Texas Instruments Incorporated., “CD74HC4067 のデータシート、製品情報、およびサポート | TI.com”. https://www.ti.com/product/ja-jp/CD74HC4067, (最終アクセス 2022-12-17)
[6] J. Adams, “SCHEMATIC1 : REDUCED SCHEMATICS”. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf, (最終アクセス 2022-12-17)