卒 業 研 究 報 告 書

題　目　　Arduinoプロジェクトにおける
Example Sketchの再利用分析

指導教員　　水野　修　教授

崔　恩瀞　助教

京都工芸繊維大学 工芸科学部 情報工学課程

学生番号　　17122044

氏　　名　　寺村　英之

令和3年2月12日提出

Arduino プロジェクトにおける Example Sketch の再利用分析

令和 3 年 2 月 12 日　　17122044　寺村　英之

概　要

ソフトウェア開発では既存のソフトウェアからソースコードを再利用することがある．再利用部分はライブラリ化できる機能を含むことがある他，再利用先のソフトウェアの品質に影響を与えうる．したがってソフトウェアの再利用部分の調査はソフトウェア工学上の関心が高い．この研究では再利用部分の調査を Arduino でも可能にするために Arduino に特化した再利用検出手法を提案する．提案する手法は既存のコードクローン検出手法と分析対象を削減する手法を組み合わせたもので，実際にスケッチから再利用箇所をいくつか検出できた．さらに検出されたコード片について議論し再利用されているコードの特徴と提案手法の課題，そして Arduino の Example Sketch の効果について議論した．

目　次

1. 緒言

ソフトウェア開発では既存のソフトウェアのソースコードからその一部を再利用することがある．先行研究ではオープンソースソフトウェア開発者へのインタビューと既存の研究をもとに調査した結果，開発者は新たな機能の実装時に平均して30%の割合でコードを再利用していることが分かった [1]．ソフトウェアの再利用のされ方を調べることでソフトウェア工学での研究やソフトウェア開発において有用な情報を得ることができる．例えば開発者が自分のプロジェクト内でのコードの再利用箇所を調査すればよく使われている部分をライブラリ化や機能が似ているモジュールの統合にその情報を使うことができる．また “A brief overview of software reuse and metrics in software engineering”[2] の第 7 章では再利用のメトリクスによって Reuse Library の評価を行えることが指摘されている．Reuse Library とは再利用可能なコードを管理し，開発者によるコードの再利用を補助するシステムである．例えば Reuse Library を使ってあるコードがどれだけ再利用されているかを調べることによって，その部分のコードの品質を評価できる [3].

この研究では再利用分析を行うソフトウェアとして Arduino で開発者が開発するソフトウェアに注目する．Arduino はプロトタイピングや組み込みシステムの開発などに利用されている身近な電子工作プラットフォームの 1 つである．例えば Arduino を用いた農業分野における IoT デバイスの開発について述べたいくつかの研究がある [4]．Arduino におけるユーザは C++言語を拡張した Arduino 言語を用いてソフトウェアを開発する．Arduino 言語は専用のエントリポイントとなる特殊な関数 2 つと入出力制御などを行ういくつかの組み込み関数とデータ構造を事前に定義してあり，ハードウェアによる違いを吸収するように設計されている．したがって組込みシステムでの開発の知識がなくても容易に開発できる．このようにユーザが開発するソフトウェアは Sketch と呼ばれる．Sketch は Arduino ボードを PC に接続して IDE から 1 クリックでダウンロードするだけで動作させることができ，LED ランプなどの電子回路をその場で動作させながら開発できる．また，ボードに接続する電子回路は自作のものだけでなく既に組み立てられている Shield と呼ばれる基板を繋いで使うこともできる．これによって電子回路に詳しくなくても複雑な装置を組み立てることが容易になっている．

電子工作やプロトタイピングだけでなく研究や産業目的にArduinoが利用されるケースもある．例えばCERNではArduinoを利用したシステムで実験装置を作成している[5]．また，Arduinoの産業的な利用を目的とするArduino Proがリリースされ，産業分野での利用例がある[6]．このようにArduinoの応用はプロトタイピングの範囲を超えた広がりを見せている．

Arduinoにおけるソフトウェアの開発を高速化しプロトタイピングや実世界での利用においてArduinoを有利にしている仕組みとして考えられるものの1つとしてLibraryと呼ばれる仕組みかある．Arduinoにおける Libraryとは電子工作でよく使われるハードウェアやShieldの制御プログラム，およびJSONパーサなどの再利用可能な処理を実装したソースコード群である．Libraryは Arduino IDEから容易にキーワード検索できる．開発者は必要なライブラリを検索してダウンロードし自身のSketchにインクルードすることでその機能を利用できる．多くのLibraryにはExample Sketchがいくつか同梱されている．これらはそのLibraryの利用例をSketchとして表したものである．それらを開発者のボードにダウンロードすることでライブラリや接続した回路の機能をコーディングなしに評価できる．また，Example Sketch自体を再利用することで開発者が実装したい機能を容易に実現できる．

このようにArduinoは高速な開発を必要とするプロトタイピングやソフトウェアの信頼性を要求する実世界での応用に利用されている他，コードの再利用を容易にする仕組みを備えている．一般的なソフトウェア開発をはじめ，特に信頼性が求められるものではコードを再利用する場合のソフトウェアの品質管理が重要である．例えば開発者が再利用箇所と再利用元を把握できれば再利用部分に関係する問題が発生した場合に迅速に修正できる．またソフトウェア工学の分野に置いてはArduinoにおけるExample Sketchをある種のReuse Libraryと考えられる．様々なスケッチでExample Sketchがどれほど再利用されているかを調べればReuse Libraryのアセットとしての Example Sketchの質について議論できる．したがってArduinoにおけるコードの再利用の調査はソフトウェア工学の研究とArduinoプロジェクトの開発の両方に利点がある．しかし，我々の調査した限りではArduinoにおけるソフトウェア開発に注目したコードの再利用の調査についての研究を見つけることができなかった．

この研究ではArduinoにおけるソフトウェア開発でのコードの再利用の調査を目的として既存の手法を組み合わせた調査方法の検討と小規模な再利用調査をした．

特に Arduino における再利用の仕組みの 1 つである Library の Example Sketch からのコードの再利用に注目した．この研究における研究設問を以下に示す．

RQ1 Arduino における Example Sketch からのコードの再利用を既存の方法を応用して検出するにはどうすればよいか．

RQ2 再利用されているコードの特徴はなにか．

この研究の貢献は次の 3 つである．

- Arduino における Example Sketch からのコードの再利用を既存の手法を応用して検出した．
- その手法を用いて実際に検出された再利用されていると考えられるコードの特徴について議論した．
- 検出されたコード片からの情報の有用性を確かめるために使われていたライブラリの Example Sketch の有用性について議論した．

最後にこの論文の章構成について簡潔に述べる．第 2 章ではこの研究の関心の対象である Arduino とソースコードの分析に利用するコードクローンについて必要な説明をする．第 3 章ではコードの再利用を既存の手法を応用して検出する方法を提案する．続いて第 4 章でその手法を用いた再利用の検出試行の条件を述べ，その結果を第 5 章に示す．その後第 6 章で研究設問に答えるための議論をする．最後に第 7 章で妥当性への脅威について述べ，第 8 章でこの研究の背景と貢献についてまとめる．

2. 背景

2.1 Arduino

Arduinoがどのようなものであるかは緒言で簡潔に説明した．ここではArduinoでのスケッチの開発とExample Sketchの使われ方について補足する．

Arduinoで開発するプログラムはスケッチと呼ばれる．スケッチは一般的なIDEでいうプロジェクトと対応する．スケッチには唯一のスケッチファイルと任意の数のCあるいはC++ソースコードやヘッダファイルが含まれる．作成したスケッチはArduino IDEでコンパイルしArduinoボードにアップロードするとボード上で動作させられる．

開発者は必要に応じてライブラリを自分のスケッチで使うことができる．Arduinoでいうライブラリとは電子工作でよく使われるハードウェアやシールドの制御プログラム，およびJSONパーサなどの再利用可能な処理を予め実装したものである．ライブラリのヘッダファイルをスケッチのソースコードでincludeするとその機能を使うことができる．ライブラリはArduino IDEから検索でき，必要なinclude文の追加も自動で行われる．

Arduinoライブラリはその使い方を説明するためのExample Sketchを含むことがある．多くのライブラリでそれが提供する機能や対応しているハードウェアの種類ごとに分類されたスケッチが用意されておりIDEから簡単に一覧表示でき，またボードにアップロードできる．このようなスケッチをライブラリのExample Sketchと呼ぶ．例えばほとんどのArduinoボードで利用可能なキャラクタ液晶制御ライブラリであるLiquidCrystalライブラリはHelloWorldというExample Sketchファイルを同梱している．このスケッチは液晶画面に固定メッセージと経過時間を表示するサンプルプログラムとなっている．

Example Sketchはライブラリの利用時に頻繁に使うコードをしばしば含んでおりそこからの再利用はスケッチの開発においてよく行われる．例えばEthernetライブラリのWebClientというExample SketchにはEthernetシールドの初期化処理とLANへの接続処理が含まれている．Ethernetシールドを使いたい開発者はこの部分をコピーして自分のスケッチで簡単にシールドを使い始めることができる．

2.2 コードクローン

次に，コードクローンの定義とこの研究で利用したコードクローン検出手法である CCFinderSW[7] について説明する．

肥後らの論文には「コードクローンとは, ソースコード中に存在する互いに一致または類似したコード片を指す.」[8] とある．例えば同じ C ソースコードファイル内に出現する仮引数の種類が同じ関数プロトタイプ宣言やソースファイル間で再利用されたコード片はコードクローンであると言える．

あるコード片がコードクローンであるためにはコード片同士が必ずしも完全に一致する必要はない．例えばある配列に連続して値を保存する部分があるとする（リスト 2.1）．

リスト 2.1　配列にデータを保存するコード片

```
1    int array[4];
2    array[0] = 0;
3    array[1] = 1;
4    array[2] = 2;
5    array[3] = 3;
```

ここで別の名前の配列に同様の操作をしているコード片を考える．このコード片はリスト 2.1のコード片のある種のコードクローンと言える．さらにこのコード片のどこかに行が挿入されているようなコード片についても同様に考えることができる．このように完全一致するコード片だけでなく相違があるものもある種のコードクローンとして考える．コードクローンはどれだけの相違を許すかによっていくつかのタイプに分類されている．コードクローンの分類の詳細についてはこの節で引用した論文 [8] の第 2 章 2 節に説明と参考資料がある．

この論文ではコードクローンの定義として先程引用した「ソースコード中に存在する互いに一致または類似したコード片」を採用する．

次に提案手法で利用したコードクローン検出手法である CCFinderSW について説明する．

CCFinderSW はトークンベースのコードクローン検出手法の一つである．トークンベースとはソースコードの比較にプログラミング言語のトークンを単位として用

いるという意味である．CCFinderSW がコードクローンを検出する仕組みを完結に説明する．まず入力されたソースコードからコメントを除去し，予め与えられた文法定義に基づいてトークンに分割する．次にトークン列に含まれる英数字列を予約語リストを元に予約語と識別子に分ける．その後比較したいソースコードのトークン列同士で比較を行い一致している部分をコードクローンとして報告する．つまり例えば識別子や数値はその値にかかわらず同じものとして扱われるのでソースコード内の変数名や定数が異なっていてもコードが意味論上似ているならばコードクローンとして報告されることを意味する．CCFinderSW は Java 言語で書かれたコマンドラインツールとして実装されている．コードクローンを検出したいソースコードをディレクトリに集めて実行するだけで容易にコードクローンを検出でき，適切なスクリプトを作ればバッチ処理も可能である．

2.3 Arduino プロジェクトでの再利用分析

背景の説明の最後として Arduino プロジェクトでの Example Sketch の再利用分析の必要性について述べる．

まず，実世界にあるスケッチについてどの Example Sketch が再利用されているのかを知ることで Example Sketch がスケッチ開発にどれほど影響を与えているか調べられる．例えば似た機能を実装している複数のスケッチに対してその調査することである機能を実現することに関連したライブラリの機能を知ることができる．

さらに，スケッチの開発者が自分のスケッチを分析することで利用していたライブラリや再利用した Example Sketch を知ることができる．これによりどのライブラリを使っていたか記録されていなかった場合にその情報を復元できる可能性がある．また，スケッチ内の Example Sketch からの再利用部分を管理することにも役立つ．

このように一般的なソフトウェアでの再利用分析の利点だけでなく，Arduino における開発特有の利点もある．

3. 提案手法

3.1 コードクローンの検出

Example Sketch のコードからの再利用はコードクローンとして検出できる．例えばある Example Sketch のコードの一部がスケッチにコピーされているとする．このときその Example Sketch とスケッチとの間のコードクローンは該当再利用箇所を含む可能性がある．また，スケッチがある Example Sketch をもとにして作られている場合も大きく変更が加えられなかった部分はコードクローンとなる．したがって両者の間のコードクローンを検出しその結果を調べることで再利用されたコードを知ることができる．

既存の研究では様々なコードクローン検出手法が提案されている．この研究では検出手法を次の 3 つの手法の中から選択した．それぞれの手法についての詳細は引用されている論文を参照されたい．

- Deckard[9]
- CCFinderSW[7]
- SourcererCC[10]

検出手法を選択する上で重要な要素は 2 つある．1 つは C++言語に対応していること，もう 1 つは関数単位，あるいはそれよりも細かい粒度でクローン箇所を検出できることである．スケッチの主要なソースコードであるスケッチファイルとそれに付随するソースコードはすべて Arduino 言語で書かれている．そして Arduino 言語は C++言語をもとにしているので検出手法が C++言語に対応していることが重要となる．また，Example Sketch からの再利用ではファイル単位よりも小さなクローンが多く発生すると考えることができる．なぜならば開発者は Example Sketch を再利用した結果それとほとんど同じ Sketch を作成するとは考えにくいからである．したがって Example Sketch からの再利用の様子を知るためには検出の粒度がある程度細かくなくてはならない．

検討を開始した時点で C++言語のサポートがあり，かつ細かい粒度でクローン箇所を検出できる手法は CCFinderSW のみだった．したがって CCFinderSW をこれら

の中から選択した．

CCFinderSW でソースコード間のコードクローンを見つけるためには対象となるソースコードをすべて入力とすれば良い．したがって，スケッチと Example Sketch との間のコードクローンを見つけるにはそれらのソースコードをすべて入力すれば良い．しかし，これだけでは 1 つのスケッチに対してかなり時間がかかってしまう．まず，ライブラリの種類は IDE からデフォルトの設定で検索できるものだけでも 3560 ほどある．それぞれのリリースされたバージョンも合わせれば全体で 16000 種類ほどになる．あるスケッチで使われるライブラリはこの数よりもかなり少ない上，実際に参照された Example Sketch も一部にとどまると考えられる．したがって，この全てを入力とすると多くの Example Sketch に対する処理が無駄になる．また，CCFinderSW は入力されたソースコードからペアを作り，その全てに対してコードクローンを検出する．ところが，この研究で注目したいものは Example Sketch とスケッチの間のコードクローンである．このまま入力すれば Example Sketch 同士の分析も行われてしまうのでほとんどの処理が無駄になる．そこで CCFinderSW に適切な入力を行い分析するためのツールを作成した．

3.2 作成したツールの説明

前の節では単に Example Sketch とスケッチを入力とすると次の 2 つの問題が起こることを説明した．

1. スケッチで使われていない Example Sketch についても分析が行われる
2. Example Sketch 同士についても分析が行われる

ここでは作成したツールで以上の問題をどのように解決したかを説明する．

問題 1はスケッチと関係があると考えられる Example Sketch を絞り込むことで解決した．まず，次にライブラリごとにライブラリのソースコードであるヘッダファイルの集合を作る．次に，すべての Example Sketch のスケッチファイルについてその中で include されているヘッダファイルの集合を作る．その後 Example Sketch のヘッダ集合と対応するライブラリのヘッダファイル集合の積をとり，その Example Sketch の特徴ヘッダファイル集合とする．スケッチから Example Sketch を絞り込むには次

のようにする．まず，スケッチのスケッチファイルで include されているヘッダファイルの集合を作る．その後すべての Example Sketch の特徴ヘッダファイル集合についてそれがスケッチのヘッダファイル集合に含まれているか確かめる．もし含まれているならば対応する Example Sketch を分析対象に入れる．特徴ヘッダファイル集合はスケッチから利用できるライブラリのヘッダファイルのうち Example Sketch で使われているヘッダファイルの集合である．そのヘッダファイルの集合がスケッチで include されているということは対象の Example Sketch で使われている機能が使われている可能性があることを意味する．これによってスケッチで使用されようとしている機能に最も近い Example Sketch を選ぶことができる．

問題 2はスケッチと Example Sketch のソースコードを予めペアにして入力とすることで解決した．まず，スケッチと Example Sketch のソースコードをそれぞれ 1 つずつペアにしたすべての組み合わせを作成する．次に，それらのペアごとにディレクトリを作りそれぞれのソースコードを格納する．そして作成したすべてのディレクトリを入力としてクローン検出ツールを実行し結果を収集する．これによって比較する意味のないペアを除いて分析できる．

表 4.1　CCFinderSW のパラメータ

パラメータ	値
対象の言語 (-l)	cpp
クローンペアの範囲 (-w)	2（ファイル間のみ）
-antlr	”ino|pde|c|h|cpp|hpp|cxx|hxx|cc”
文字コード (-charset)	auto

4. 検出試行の条件

前の章で説明した手法を用いて実際に再利用部分を検出できるか試行した．この章では試行時の各種条件について述べる．

再利用を検出するスケッチとして big_aquatan(注 1)を用いた．検出対象のライブラリ群として Arduino IDE のデフォルトライブラリメタデータ(注 2)にあるものを用いた．メタデータとライブラリのダウンロードはどちらも 2021 年 1 月 4 日から 1 月 5 日のうちに行った．ライブラリは計算リソースの都合上ダウンロードした時点で最新バージョンのものだけを検出対象にした．CCFinderSW のバージョンは 1.0 を用い，CCFinderSW のパラメータは表 4.1のように設定した．

この表に記述がないものは入力ディレクトリと出力ファイル名に関わるオプションである”-d”と”-o”を除いてすべてデフォルト値を指定した．また，Arduino 言語のファイルも分析対象に入れるため，grammarsv4 ディレクトリに拡張子”ino”，”pde” のための文法定義を追加した．文法ファイルは C++言語のために予め用意されている CPP14.g4 を流用した．

検出試行は Example Sketch の絞り込みを行った場合と行わなかった場合の両方行った．

(注 1)：https://github.com/omzn/big_aquatan/tree/ed1e327f3ff307e5fe1a7f9c6fe32874c2fc36ea
(注 2)：https://downloads.arduino.cc/libraries/library_index.json

5. 検出試行の結果

検出試行の結果 Example Sketch を絞り込んだ場合は 5 ペア，絞り込まなかった場合は 86 ペアのソースコードからコードクローンが検出された．この結果からさらにコードが実装している機能をもとにコードクローンを分類した．それぞれの条件に分けて以下で述べる．

5.1 Example Sketch を絞り込んだ場合

この条件で検出されたクローンはすべて SparkFun APDS9960 RGB and Gesture Sensor ライブラリバージョン 1.4.2 を使った初期化コードだった．特にそのうち 1 つのコードクローンはジェスチャ検出機能を有効化する部分が含まれていた．big_aquatan での該当箇所（リスト 5.1）とジェスチャ検出機能を有効化する部分が含まれている結果の該当箇所（リスト 5.2），およびそれ以外の結果の代表的な箇所（リスト 5.3）を示す．

リスト 5.1　スケッチの big_aquatan.ino の該当箇所

```
127   Wire1.begin(PIN_SDA1, PIN_SCL1);
128
129   eyes.begin(-5, 0, 0, 0);
130
131   if (apds.init()) {
132     Serial.println(F("LAPDS-9960 initialization complete"));
133   } else {
134     Serial.println(F("Something went wrong during APDS-9960 init!"));
135   }
136   if (apds.enableGestureSensor(true)) {
137     Serial.println(F("Gesture sensor is now running"));
138   } else {
139     Serial.println(F("Something went wrong during gesture sensor init!"));
140   }
```

リスト 5.2　Example Sketch の GestureTest/GestureTest.ino の該当箇所

```
71
72    // Initialize interrupt service routine
73    attachInterrupt(0, interruptRoutine, FALLING);
74
75    // Initialize APDS-9960 (configure I2C and initial values)
76    if ( apds.init() ) {
77      Serial.println(F("APDS-9960 initialization complete"));
78    } else {
79      Serial.println(F("Something went wrong during APDS-9960 init!"));
80    }
81
82    // Start running the APDS-9960 gesture sensor engine
83    if ( apds.enableGestureSensor(true) ) {
84      Serial.println(F("Gesture sensor is now running"));
85    } else {
86      Serial.println(F("Something went wrong during gesture sensor init!"));
87    }
88  }
```

リスト 5.3　Example Sketch の ColorSensor/ColorSensor.ino の該当箇所

```
53    Serial.println(F("----------------------------------"));
54    Serial.println(F("SparkFun APDS-9960 - ColorSensor"));
55    Serial.println(F("----------------------------------"));
56
57    // Initialize APDS-9960 (configure I2C and initial values)
58    if ( apds.init() ) {
59      Serial.println(F("APDS-9960 initialization complete"));
60    } else {
61      Serial.println(F("Something went wrong during APDS-9960 init!"));
62    }
63
64    // Start running the APDS-9960 light sensor (no interrupts)
65    if ( apds.enableLightSensor(false) ) {
66      Serial.println(F("Light sensor is now running"));
```

SparkFun APDS9960 RGB and Gesture Sensor ライブラリ(注 1)は APDS-9960(注 2)センサの制御ライブラリである．このライブラリはセンサの初期化とジェスチャ認識などの各種機能の有効化機能を実装している．big_aquatan スケッチは APDS-9960 センサを用いてハンドジェスチャを認識する機能がある． また，該当部分のメッセージ文字列やコードの書式がよく似ており，これらから再利用として適切な部分が検出されたと言える．

これらの部分は 5 個のソースコードのペアから見つかり，範囲が重なっているコードクローンを含めて全部で 5 個だった．

5.2 Example Sketch を絞り込まなかった場合

この条件で検出されたクローンは次の種類に分けられた．

1. SparkFun APDS9960 RGB and Gesture Sensor ライブラリを使った初期化コード

2. リアルタイムクロックの時刻取得コード

3. SNTP 関連のコード

4. arduino-esp32 の OTA イベント処理コード

5. 再利用ではないコード

種類 1のコードクローンの数は 5 つだった．これは Example Sketch を絞り込んだ場合に検出されたものだと考えられる．なぜならこの条件で比較した Example Sketch の中に絞り込んだ場合の Example Sketch が全て含まれている上，検出されたクローンが同じ 5 箇所だったからである．

種類 2のコードクローンは主にリアルタイムクロックから取得した時刻を書式化する処理を実装していた．big_aquatan での該当箇所（リスト 5.4）と Example Sketch での該当箇所の代表（リスト 5.5）を示す．

(注 1): https://github.com/sparkfun/APDS-9960_RGB_and_Gesture_Sensor/
(注 2): https://www.broadcom.com/products/optical-sensors/integrated-ambient-light-and-proximity-senso

リスト 5.4　スケッチの big_aquatan.ino の該当箇所

```
653   DateTime now = rtc.now();
654   char str[20];
655   sprintf(str, "%04d-%02d-%02d %02d:%02d:%02d", now.year(), now.month(),
656           now.day(), now.hour(), now.minute(), now.second());
657   ts = str;
658   return ts;
```

リスト 5.5　MCP7840-1.2.0 の SetAndCalibrate/SetAndCalibrate.ino の該当箇所

```
173   DateTime        now = MCP7940.now();   // get the current time
174   if (secs != now.second()) {            // Output if seconds have changed
175     sprintf(inputBuffer, "%04d-%02d-%02d %02d:%02d:%02d",
176             now.year(),  // Use sprintf() to pretty print
177             now.month(), now.day(), now.hour(), now.minute(),
178             now.second());                          // date/time with
                ↪ leading zeros
179     Serial.println(inputBuffer);                    // Display the current
        ↪ date/time
180     secs = now.second();                            // Set the counter
        ↪ variable
```

コードクローンを含む Example Sketch に対応するライブラリはすべてリアルタイムクロックの制御ライブラリだった．これらの Example Sketch も多少の改変を除けば現在時刻を取得し文字列に初期化する処理を含んでいた．しかし再利用であると考えにくい部分もある．big_aquatan では sprintf 関数の行の改行位置が now.day メソッドの呼び出し直前にある．それに対し Example Sketch では now.year や now.hour メソッド呼び出しの直前で改行されていた．また書式文字列は ISO8601 に基づいた日時表現とよく似ているほか，現在時刻のアクセス部分も対応するメソッドを呼び出すだけで実現できるので特殊なものであるとは言えない．したがってこの部分が再利用であるかどうかは判断できなかった．

これらの部分は 8 個のソースコードのペアから見つかり，範囲が重なっているコードクローンを含めて全部で 10 個だった．

種類 3のコードクローンは 2 種類に分類できた．1 つは SNTP パケットを作成する

コード，もう 1 つは SNTP サーバから受信したデータをもとに 1900 年からの経過秒数を計算するコードだった．big_aquatan での該当箇所（リスト 5.6，5.8）と Example Sketch での該当箇所の代表（リスト 5.7，5.9）を示す．

リスト 5.6　スケッチの ntp.cpp の SNTP パケット作成部分

```
10
11 // send an NTP request to the time server at the given address
12 void NTP::sendPacket() {
13
14   // set all bytes in the buffer to 0
15   memset(packetBuffer, 0, NTP_PACKET_SIZE);
16   // Initialize values needed to form NTP request
17   // (see URL above for details on the packets)
18   packetBuffer[0]  = 0b11100011;   // LI, Version, Mode
19   packetBuffer[1]  = 0;     // Stratum, or type of clock
20   packetBuffer[2]  = 6;     // Polling Interval
21   packetBuffer[3]  = 0xEC;  // Peer Clock Precision
22   // 8 bytes of zero for Root Delay & Root Dispersion
23   packetBuffer[12] = 49;
24   packetBuffer[13] = 0x4E;
25   packetBuffer[14] = 49;
26   packetBuffer[15] = 52;
27   // all NTP fields have been given values, now
28   // you can send a packet requesting a timestamp:
29   udp.beginPacket(timeserver.c_str(), 123); //NTP requests are to port 123
30   udp.write(packetBuffer, NTP_PACKET_SIZE);
```

リスト 5.7　RTCDue-1.1.0 の RTCDue_NTP_WIFI_with_timezone/RTCDue_NTP の該当箇所

```
183
184 // send an NTP request to the time server at the given address
185 unsigned long sendNTPpacket(IPAddress& address){
186   // set all bytes in the buffer to 0
187   memset(packetBuffer, 0, NTP_PACKET_SIZE);
188   // Initialize values needed to form NTP request
189   // (see URL above for details on the packets)
190   packetBuffer[0]  = 0b11100011;   // LI, Version, Mode
```

```
191   packetBuffer[1]  = 0;            // Stratum, or type of clock
192   packetBuffer[2]  = 6;            // Polling Interval
193   packetBuffer[3]  = 0xEC;         // Peer Clock Precision
194   // 8 bytes of zero for Root Delay & Root Dispersion
195   packetBuffer[12] = 49;
196   packetBuffer[13] = 0x4E;
197   packetBuffer[14] = 49;
198   packetBuffer[15] = 52;
199
200   // all NTP fields have been given values, now
201   // you can send a packet requesting a timestamp:
202   Udp.beginPacket(address, 123); //NTP requests are to port 123
203   Udp.write(packetBuffer, NTP_PACKET_SIZE);
```

リスト 5.8　スケッチの ntp.cpp の受信データ処理部分

```
37   unsigned long secsSince1900 = 0;
38   // convert four bytes starting at location 40 to a long integer
39   secsSince1900 |= (unsigned long)packetBuffer[40] << 24;
40   secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
41   secsSince1900 |= (unsigned long)packetBuffer[42] <<  8;
42   secsSince1900 |= (unsigned long)packetBuffer[43] <<  0;
43   return secsSince1900 - 2208988800UL; // seconds since 1970
```

リスト 5.9　log4Esp-1.0.1 の AdvancedDemo/AdvancedDemo.ino の該当箇所

```
137       unsigned long secsSince1900;
138       // convert four bytes starting at location 40 to a long integer
139       secsSince1900 = (unsigned long)packetBuffer[40] << 24;
140       secsSince1900 |= (unsigned long)packetBuffer[41] << 16;
141       secsSince1900 |= (unsigned long)packetBuffer[42] << 8;
142       secsSince1900 |= (unsigned long)packetBuffer[43];
143       return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR;
```

SNTP パケットを作成する部分は識別子や周辺のコメントなどがよく一致しており，再利用されたコードであると判断して問題ないと言える．

一方で受信したデータを処理するコードはスケッチのものと Example Sketch のも

のの間で違いが見られた．big_aquatan のコードでは Example Sketch のコードで実装されているタイムゾーンに合わせて秒数を調整するコードが失われている．また packetBuffer[43] の直後の 0 ビットシフトは Example Sketch のコードには存在しなかった．それでもスケッチと Example Sketch のコード片には識別子などの共通点が見られるので，共通の源流となるコードを持つなど何らかの関係があると考えられる．

これらの部分は 55 個のソースコードのペアから見つかり，範囲が重なっているコードクローンを含めて全部で 171 個だった．このことから検出されたコードクローンの部分は様々なライブラリの Example Sketch で再利用されていることがわかる．

種類 4のコードクローンは OTA のエラーイベント処理を実装していた．big_aquatan での該当箇所（リスト 5.10）と Example Sketch での該当箇所の代表（リスト 5.11）を示す．

リスト 5.10　スケッチの big_aquatan.ino の該当箇所

```
162       .onProgress([](unsigned int progress, unsigned int total) {
163         digitalWrite(2, (progress / (total / 100)) % 2);
164         Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
165       })
166       .onError([](ota_error_t error) {
167         Serial.printf("Error[%u]: ", error);
168         if (error == OTA_AUTH_ERROR)
169           Serial.println("Auth Failed");
170         else if (error == OTA_BEGIN_ERROR)
171           Serial.println("Begin Failed");
172         else if (error == OTA_CONNECT_ERROR)
173           Serial.println("Connect Failed");
174         else if (error == OTA_RECEIVE_ERROR)
175           Serial.println("Receive Failed");
176         else if (error == OTA_END_ERROR)
177           Serial.println("End Failed");
178       });
179   ArduinoOTA.begin();
180
181   webServer.on("/", handleStatus);
182   webServer.on("/reboot", handleReboot);
```

リスト 5.11　IRremoteESP8266-2.7.14 の IRrecv-

DumpV3/BaseOTA.h の該当箇所

```
39      })
40      .onProgress([](unsigned int progress, unsigned int total) {
41        Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
42      })
43      .onError([](ota_error_t error) {
44        Serial.printf("Error[%u]: ", error);
45        if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
46        else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
47        else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed")
            ↪ ;
48        else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed")
            ↪ ;
49        else if (error == OTA_END_ERROR) Serial.println("End Failed");
50      });
51
52    ArduinoOTA.begin();
53    Serial.println();
54    if (WiFi.waitForConnectResult() == WL_CONNECTED) {
```

OTA とはネットワーク上にあるデバイスのソフトウェアのアップデートを可能にする機能である．Arduino の場合は ArduinoOTA ライブラリでその機能を実現している．検出された部分では”ArduinoOTA”という識別子が出現するのでこの機能に関係があると考えることができる．ところが ArduinoOTA ライブラリ(注 3)の Example Sketch と検出部分を比較したところ，検出された部分は ArduinoOTA には存在しなかった．更に調査したところ検出された部分と似た部分を持つ Example Sketch が複数見つかった．

- arduino-esp32 の BasicOTA(注 4)
- ESP8266/Arduino の BasicOTA(注 5)

これらはそれぞれ ESP32 シリーズと ESP8266 マイクロコントローラ向けの Arduino ソフトウェアの実装に含まれる Example Sketch である．実際，検出されたコードク

(注 3): https://github.com/jandrassy/ArduinoOTA
(注 4): https://github.com/espressif/arduino-esp32/blob/master/libraries/ArduinoOTA/examples/BasicOTA/
(注 5): https://github.com/esp8266/Arduino/blob/master/libraries/ArduinoOTA/examples/BasicOTA/BasicOTA

ローンを含む Example Sketch に対応するライブラリはすべて ESP32 か ESP8266 を用いたシステムを前提としていた．したがってこれらの Example Sketch でもコードが再利用されたと考えられる．

クローンペアを観察するとエラーメッセージやエラーコードを確認する際の比較の順番などが一致しており，検出された部分は再利用されている可能性があると考えられる．特に big_aquatan の該当部分は ESP8266/Arduino の BasicOTA にある部分とよく似ていた．したがって該当部分はその Example Sketch の実装を源流としている可能性がある．

これらの部分は 6 個のソースコードのペアから見つかり，範囲が重なっているコードクローンを含めて全部で 30 個だった．

種類 5のコードクローンは再利用ではないと考えられるもので，2 種類に分類できた．

- 関数プロトタイプ宣言とクラスメソッド宣言
- 配列への値の保存

検出例をそれぞれリスト 5.12，5.13とリスト 5.14に示す．Example Sketch の配列への値の保存部分はすべてスケッチの SNTP パケット作成部分（リスト 5.6）とペアになっていた．

リスト 5.12　スケッチの actionqueue.h の該当箇所

```
34 public:
35   actionQueue(AquatanEyes *e, AquatanArms *a, Camera *h,
36               NeoPixels *c); //, Charger *c);
37   void queueEyes(int l_shape, int r_shape, int m);
38   void queueArms(int left, int right, int time);
39   void queueArmLeft(int left, int time);
40   void queueArmRight(int right, int time);
41   void queueHead(int pan, int tilt);
42   void queueHeadPan(int pan);
43   void queueHeadTilt(int tilt);
```

リスト 5.13　ES920-0.1.9 の openFrameworks/ascii/src/ofApp.h の該当箇所

```
13                  void keyReleased(int key);
14                  void mouseMoved(int x, int y );
15                  void mouseDragged(int x, int y, int button);
16                  void mousePressed(int x, int y, int button);
17                  void mouseReleased(int x, int y, int button);
18                  void mouseEntered(int x, int y);
19                  void mouseExited(int x, int y);
20                  void windowResized(int w, int h);
21                  void dragEvent(ofDragInfo dragInfo);
22                  void gotMessage(ofMessage msg);
```

リスト 5.14　Esp32SimplePacketComs-0.7.0 の SwarmRobotExample/SwarmRobotExample.ino の該当箇所

```
13          // User data is written into the buffer to send it back
14          void event(float * buffer) {
15                  int numberOfFloats = 15;
16                  buffer[0] = 10; // X Location
17                  buffer[1] = 0;  // Y Location
18                  buffer[2] = 0;  // Z Location
19                  buffer[3] = 45; // azimuth
20                  buffer[4] = 0;  // elevation
21                  buffer[5] = 0;  // tilt
22                  buffer[6] = 100;        // X size
23                  buffer[7] = 100;        // Y size
24                  buffer[8] = 100;        // Z size
25  //              Serial.println(
```

big_aquatan のソースコードに含まれる関数プロトタイプ宣言やクラスメソッド宣言は Example Sketch の名前にある異なる関数とペアとして検出された．big_aquatan スケッチにあったこれらの関数プロトタイプ宣言やクラスメソッド宣言はすべて独自の機能に関係するものだと考えられるので再利用ではないと言える．

また，ntp.cpp に含まれる SNTP パケット作成処理の部分と Example Sketch で見られる様々な配列への代入部分もコードクローンとして報告されていた．これらは処

理の意味が異なっており再利用ではないと考えられる.

これらの部分は 12 個のソースコードのペアから見つかり，範囲が重なっているコードクローンを含めて全部で 39 個だった.

6. 議論

この章では前の章の結果に基づいて議論し，研究設問に回答する．

6.1 再利用の検出手法

まず研究設問 1 に回答する．

再利用の検出試行では提案手法を用いて再利用と考えられるいくつかのコード片を発見できた．Example Sketch の絞り込みのありなしにかかわらず比較は 3 日以内に完了し，再利用ではない検出結果も少なかった．したがってこの手法は現実的なリソースで Example Sketch からの再利用を探すことに一定の効果があると言える．

一方で提案した Example Sketch の絞り込みが再利用されたコードの発見を妨げていることもわかった．絞り込みによって発見できなかった再利用の可能性がある部分は次の 3 つである．

- リアルタイムクロックの時刻取得コード
- SNTP パケット送信コード
- arduino-esp32 の OTA イベント処理コード

まずリアルタイムクロックの時刻取得コードがなぜ検出できなかったかについて議論する．Example Sketch を絞り込んだときリアルタイムクロック関係のものとして RTClib の Example Sketch が対象に含まれていた．しかし RTClib の Example Sketch には発見されたような書式化処理はなかった．つまりヘッダファイルの共通性から Example Sketch を絞り込むと似たような機能を持つ異なるライブラリからの再利用を検出できないと考えられる．

次に SNTP パケット送信コードがなぜ検出できなかったについて議論する．このコードは UDP 通信機能を使っているあるいは提供しているライブラリの Example Sketch にあった．例えば log4Esp の AdvancedDemo では WifiUdp.h が include されており，WiFi ライブラリの UDP 通信機能が SNTP 通信のために使われている．一方 big_aquatan ではスケッチファイルでなく ntp.h の中で include されていたため絞り込みの際にそのヘッダを考慮に入れることができていなかった．したがってどのヘッダ

が include されているのかを絞り込みの情報に使うことはそのままでは難しく，ヘッダファイル内の include プリプロセッサ命令まで考慮する必要があるとわかった．

arduino-esp32 の OTA イベント処理コードが検出できなかったことは他の要因が関わっている．ライブラリをダウンロードする際ライブラリメタデータに登録されているものをダウンロードした．しかし，これらには Arduino コアが提供するライブラリは含まれていない．Arduino コアとは Arduino ボードやマイクロコントローラの種類ごとに実装されている中核となるソフトウェアである．これにはボードに書き込まれるブートローダや入出力など基本的な操作を提供するコード，そしていくつかのライブラリが含まれている．コアに含まれるライブラリは基本的な機能を提供する少数のライブラリとボード特有の機能を提供するライブラリがある．arduino-esp32 の ArduinoOTA ライブラリや WiFi ライブラリもこのライブラリの一種である．これらのライブラリの Example Sketch も考慮に入れたい場合は Arduino コアのライブラリからも Example Sketch を予め集めておく必要があると言える．

6.2 再利用の特徴

次に研究設問 2 に回答するため，前の章の発見できた再利用コード片の分析をまとめる．

ここでは再利用であるか判断できなかったリアルタイムクロックの時刻取得コードを省いた次の 3 種類の再利用されているコードの特徴について議論する．

- SparkFun APDS9960 RGB and Gesture Sensor ライブラリを使った初期化コード
- SNTP パケット送信コード
- arduino-esp32 の OTA イベント処理コード

APDS9960 の初期化コードと OTA イベント処理コードはエラーの提示方法を工夫するなどの動機がない限り改変する必要のないコードだった．また，SNTP パケットの作成についても時刻を取得するだけならば内容を工夫する必要はない．実際 Blynk ライブラリ(注 1)ではこの部分が BlynkSimpleEthernetSSL.h に取り込まれている．した

(注 1)：https://github.com/blynkkk/blynk-library

がって再利用されているコードにはある機能を実装するコードスニペットあるいは一般的なソフトウェア開発でのライブラリとも言える特徴があると考えられる．

しかしこの検出試行で発見できた再利用の種類は 3 種類にとどまった上，対象としたスケッチも 1 つだけだから一般的な Arduino における開発に一般化できない．再利用の特徴についてはさらなる調査が必要であると言える．

6.3 Example Sketch 同士の類似性

検出されたコードクローンを分析している際に同じライブラリや異なるライブラリの Example Sketch によく似た検出箇所が見られた．例えば SparkFun APDS9960 RGB and Gesture Sensor ライブラリを使った初期化コードは同ライブラリ内の複数の Example Sketch に存在した．また，ESP32 を用いるシステムを対象とした複数の異なるライブラリの Example Sketch でよく似た OTA イベント処理コードが見つかった．これらから以下の 2 つの仮説を建てることができる．

- Example Sketch 間でもコードを再利用している
- 各 Example Sketch に共通の再利用元コードが存在する

これらは origin analysis に向いている仮説だから，Example Sketch がどのように作られたかどうかを調査することで明らかにできると考えられる．

6.4 Example Sketch の有用性

big_aquatan では SparkFun APDS9960 RGB and Gesture Sensor ライブラリを使った初期化コードが再利用されていた．初期化処理を最初から実装するにはライブラリのドキュメントを参照しさらにエラー時の処理も考える必要がある．一方で再利用する場合は IDE から Example Sketch を探しその中からジェスチャ認識機能の準備と思われる場所を抜き出して動作を確認するだけで済む．その点から big_aquatan の開発において Example Sketch がある程度工数の削減に役立ったと考えられる．その他のライブラリについても同様の理由で効果があったと言える．

7. 妥当性への脅威

検出試行では計算リソースの都合上ダウンロードした時点で最新のライブラリのみ検出に使用した．したがって，もし古いバージョンにのみ見られるコード編が big_aquatan で再利用されていた場合はその部分を検出できない．このライブラリバージョンの制約は妥当性への脅威となりうる．

また，提案した手法を実現するツールは執筆者が開発し，必要と思われる部分にはテストを行った．しかし，それにもかかわらずツールにまだ未知のバグが含まれていることは否定できず，妥当性への脅威となりうる．

8. 結論

この論文では Arduino プロジェクトにおける Example Sketch からのコードの再利用の分析を目的とし，それに特化した検出手法を提案した．提案手法は既存のコードクローン検出手法に適切な入力作成手法を組み合わせたものだった．そして提案手法を用いて実際に再利用部分を発見することによって提案手法の効果を示すことができた．さらに検出結果を観察し，再利用されている部分の特徴や Example Sketch の再利用のされ方，および Example Sketch の有用性について議論した．一方で include されているヘッダファイルによる絞り込みに問題があるとわかった．また，再利用部分を見つけただけではそのコードがどこから再利用されたのか明らかにできないという課題も見つかった．

次にこの研究の今後の課題についてまとめる．試行検出の結果の分析はスケッチの開発者ではない執筆者が行った．したがって再利用元や再利用の目的を明らかにするため，開発者にインタビュー調査を行う必要がある．また提案手法に置いて入力データの絞り込みや検出後のデータの表現方法に課題があった．今後は提案手法を課題を明確化し手法の改良や結果の応用方法について研究を進めると良いと考える．

謝辞

本研究を行うにあたり研究方針の決定や先行研究の調査で助言をくださった崔　恩瀞助教および名古屋大学大学院情報学研究科の吉田　則裕准教授に心から感謝申し上げます．また研究のアイディアを考える上でのサポートとスケッチの提供をしてくださった水野　修教授に厚くお礼申し上げます．論文の執筆方法や実験データのまとめ方についてアドバイスしていただいた本学情報工学専攻近藤　将成先輩，大学生活を支えてくださった先生方および執筆者の家族と友人の感謝申し上げます．

参考文献

[1] M. Sojer and J. Henkel, “Code reuse in open source software development: Quantitative evidence, drivers, and impediments,” Journal of the Association for Information Systems, vol.11, no.12, p.2, 2010.

[2] A.L. Imoize, D. Idowu, and T. Bolaji, “A brief overview of software reuse and metrics in software engineering,” World Scientific News, vol.122, pp.56–70, 2019.

[3] W. Frakes and C. Terry, “Software reuse: metrics and models,” ACM Computing Surveys (CSUR), vol.28, no.2, pp.415–435, 1996.

[4] M. Abbasi, M.H. Yaghmaee, and F. Rahnama, “Internet of things in agriculture: A survey,” 2019 3rd International Conference on Internet of Things and Applications (IoT), pp.1–12, IEEE, 2019.

[5] F. Iacoangeli, A. Natochii, Y. Gavrikov, M. Garattini, G. Cavoto, F. Addesa, S. Montesano, and W. Scandale, “A smart adjustable inelastic nuclear interactions counter based on compact arduino control system and readout,” 2017 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), pp.1–4, IEEE, 2017.

[6] Arduino, terraSmart - Agriculture of the Future, （オンライン），入手先〈https://www.arduino.cc/pro/case-studies/terrasmart〉（参照 2021/02/09）.

[7] 瀬村雄一，吉田則裕，崔恩瀞，井上克郎，“多様なプログラミング言語に対応可能なコードクローン検出ツール ccfindersw,” 電子情報通信学会論文誌 D，vol.103，no.4，pp.215–227，2020.

[8] 芳樹肥後，真二楠本，克郎井上，“コードクローン検出とその関連技術,” 電子情報通信学会論文誌. D, 情報・システム = The IEICE transactions on information and systems (Japanese edition)，vol.91，no.6，pp.1465–1481，jun 2008.

[9] L. Jiang, G. Misherghi, Z. Su, and S. Glondu, “Deckard: Scalable and accurate tree-based detection of code clones,” 29th International Conference on Software Engineering (ICSE’07), pp.96–105, IEEE, 2007.

[10] H. Sajnani, V. Saini, J. Svajlenko, C.K. Roy, and C.V. Lopes, “Sourcerercc: Scaling

code clone detection to big-code," Proceedings of the 38th International Conference on Software Engineering, pp.1157–1168, 2016.

TBA