2023年度未踏ジュニア提案書

提案するプロジェクトのタイトル

ハードウェア抽象化レイヤを効果的に用いて移植容易性を実現したOS: A9N

提案者に関する事項

メインクリエータ（代表者）の氏名	伊組烈火
グループの場合、メンバーの氏名	

以下の入力欄は必要に応じてサイズを変更できます。図や表の挿入も推奨します。

1. 提案するプロジェクトの簡単な説明（**200**字以内）
次のページにある「2. 提案内容」に記載される内容を、簡潔にまとめてください。

例) 「暗記クッキー」は、海外に住む日本人の子どもや、日本語を学習したい外国人をターゲットにした、*WEB*ベースの漢字クイズで頑張って覚えた漢字がクッキーになって届くことでモチベーション継続を支援するシステムです。漢字クイズの結果を元に、レーザーカッターを利用してクッキーに漢字を刻印し、魔法のようにユーザーに届けることが出来ます。
「AN9」とは, オペレーティングシステムにおいて, オブジェクト指向設計論を用いたハードウェア抽象化レイヤー(HAL)の設計と, それを用いたOSの実装及びその体系化を目的とするものです. このHALにより, OSを異なるアーキテクチャに移植したり, 異なるデバイスへの対応が容易になります. また, この設計を用いてOSを自作したい人へ向けたドキュメントを作成し, 今後の学習者への手引とします.

2. 提案内容

提案内容を書くときのアドバイス*]*
フォーマットは自由です。図表や画像も使用できます。ページ数にも制限はありません。以下の項目は一例です。

どんなもの？
・おじいちゃん、おばあちゃんに説明すると想定してまとめてみましょう
・完成したときのイメージを図や画像で表現してみましょう

誰のどんな問題を解決するもの？　これができると誰がうれしい？
・誰がどんなシーンで使いますか
・必ずしもたくさんの人の役に立つ必要はありません
・役には立たないかもしれないけれど、おもしろいというのもありです

似た問題を解決している既存の手段（なんらかの方法や、製品、サービスなど）は何がある？
・可能であれば海外の事例も調べてみましょう。日本語よりたくさんの情報が見つかります

どのように作る？
・あなたが提案するもの（ソフトウェアやハードウェア）は、どんな技術やデータ、ツールを使ってどのように作っていくか、具体的に書いてください

あなた独自のアイデアは？
・既存のものと比べて、提案のユニークなところはどこでしょう
・改良した点を書いてみましょう

A9N(Abstraction)

クリーンなOS

A9Nとは,Cと C++をメインに用いて作成するマイクロカーネル**OS**です. 各レイヤーを適切に分離し, 高い保守性とコードの読みやすさを実現します. また, レイヤーの最下層には **HAL**(Hardware Abstraction Layer, ハードウェア抽象化レイヤー)を使用することによって, アーキテクチャやデバイス依存のコードをOSの核となるコードから分離し, 少ない労力で異なるハードウェアへの移植が可能となります.
これらは実装と同時にドキュメントに纏められ, 同年代である十代に向けたものとなります.

誰の役に立つのか？

このプロジェクトは, クリーンなOSの設計と実装を学びたい, 1年前の自分のような学習者に向けたものです.
また, このOSの学習者でさらに改良していき, 究極的にはLinuxを超えるシェアを持つOSにしていきたいと考えています.

A9Nの構造

Layer_4	Init	User Process		
Layer_3	Process Manager	File System	Network Server	
Layer_2	Device Drivers			
Layer_1	Kernel	Clock Task	System Task	
Layer_0	Hardware Abstraction Layer			

カーネル

Layer_1がマイクロカーネルです. この図のように, デバイスドライバやファイルシステムはカーネルに含まれておらず, 小さく信頼性が高い設計となっています. そのため, 不安定なモノは分離され, 安定したものとなります.
具体的には, 機構(Mechanism)であるプロセス管理やIPCはカーネルが提供し, どのように扱うかという方針(Policy)はユーザー空間上で実現されます.

IPC

IPC(InterProcess-Communication, プロセス間通信)はメッセージングにより実現します. システムコールである***send***と***receive***がそれらを提供します.
並列プログラミングにも有効で, 効果の高いシステムになっています.

プロセス

A9Nのプロセスは***fork***と***exec***システムコールによって作成され, プロセステーブルに保存されます.
各プロセスは, レジスタやスタックポインタ, プログラムカウンタ, PID, 優先度などを持っていて, コンテキストスイッチ時に復帰できるようになっています.

プロセススケジューリング

プロセススケジューリング機構は比較的単純で, ラウンドロビンスケジューリング+優先度スケジューリングを組み合わせた方式になります.
プロセスに割り当てられたクオンタムと実際に使用された時間の割合から優先度を求め, その優先度ごとにラウンドロビンスケジューリングでスケジュールします.
I/Oバーストのプロセスには高い優先度を割り当てることで, CPUを効率的に使用することが可能です.

抽象化

デバイスはInterfaceによって抽象化され, 基本的にはシステム内のコンポーネントは抽象のみに依存させるようにします.
これはHALも同じです. 抽象に対してプログラミングすることによってシステムから分離し, 高い移植容易性を実現します.

マイクロカーネルを選択した理由

OSを自作するとなった場合, 一番の選択肢となるのはモノリシックカーネルOSでしょう. 文献が豊富かつ高速のため, 商用OSは殆どがモノリシックカーネルだからです(ハイブリッドカーネルではありますが, モノリシックな側面が大きい).
しかし, 私はマイクロカーネルを選択しました.
理由は以下の通りです.

・カーネルを最小にすることによって, カーネル自体の理解を容易にできる

マイクロカーネルはその名の通りカーネルが小さいため, コード量が少なく処理を追いやすいです. そのため, 学習にはより適切であるという判断を下しました.
一方モノリシックなカーネルは, 肥大化し, 保守性が低くなり, 学習コストも増大していくでしょう. こうなっては初学者に分かりやすいものとは言えず, 非実用なOSとなってしまい, 学習に使用可能であるという目的が達成できません.

・高度な抽象化

OSを完全に自作しようと思い立つ少し前に, オブジェクト指向と設計論について学習しました.
これにより, 大規模な開発における拡張可能な設計と抽象化の技法についての知見を得ました.
これは実装よりも抽象度の高い層にある知識であり, 高レイヤーといえる存在です.
高レイヤーと低レイヤーを組み合わせた, より良い設計のOSを作成したいと考えるようになり, 今回の提案に至りました. そして, その条件に合致したのがマイクロカーネルとHALでした.

・**Linux**への嫉妬

Linuxは, Andrew S. Tanenbaumの著書である「Operating Systems: Design and Implementation」をLinus Tovaldsが読んだことにより作成されました. 私はこの事実を知ってから, この本の邦訳版かつ改訂版である「オペレーティングシステム: 設計と実装(通称MINIX本」)を読み, OSを学びました.
それと同時に, 「Linusにできるのなら自分にもできるはずだ」, と考えました. 現在, 商用OSを除いた場合, Linuxのシェアが世界を圧巻しています. そのポジションに立つOSを自作して, 勢力図を書き換えたいと強く思いました.
LinuxはMINIX本を基に作られましたが, ほぼモノリシックカーネルです. Linuxカーネルが最初に書かれた1992年当時, マイクロカーネルは速度の問題を抱えていました. IPCが多くなるので, そのオーバーヘッドにより当然パフォーマンスはモノリシックカーネルと比較して低下します.
しかし, 現在はコンピューターの著しい性能向上により, ある程度は無視できるようになったと思われます. 性能を多少無視しても, 安定性と保守性を重視する時代が到来したのではないでしょうか？

A9Nのデメリット

A9Nは100%良いものとはいえません. 当然大きなデメリットも存在します.

・vtableのコスト

私の構想している実装では, コンテキストスイッチ等のハードウェアに強く依存する部分や, 動的に処理を切り替えたい箇所は抽象クラスを用いてInterfaceとして切り分けます.
このようなポリモーフィズムの実現にサブタイプ多相を使用する際, 問題が存在します.
Interfaceに対応する実際の実装を呼ぶ際, 関数はvtable(仮想関数テーブル)を経由して呼び出されます. ダイレクトに呼べない分, パフォーマンスに対してコストがかかります.
ただし, コンパイラの最適化やCPUのジャンプ予想によっては高速化できます.

・マイクロカーネルのIPC問題

マイクロカーネルは構造上IPCが多くなります.
これにより, L4マイクロカーネルのように殆どをアセンブリで記述したカーネルの場合は別ですが, パフォーマンスに大きな影響を及ぼします.
しかし, 上記の通り安定性と保守性を重視する時代が到来したという考えで, ここは無視しています.

・C/C++ではなくRustで書くべきではないか

OSをRustで記述すべきではないだろうかと考えましたが, カーネルのようなハードウェアに近い部分はC/C++の方が小回りが効くことと, Rustによる開発経験がないため却下しました.

類似品

MINIX

「教育向けマイクロカーネルOS」は既に存在しています. その中でも代表的なのは, アムステルダム自由大学の教授であるAndrew S. Tanenbaumが開発したMINIXでしょう.
前述したMINIX本は, MINIXの作者であるTanenbaum自身がMINIXの理論と実装を解説したものです.
実際, 私もこの本でOSを学習しましたが, 現代においては問題点が存在します.
最も大きな問題点は, 内容が古くなっていることです. この本は2005年に出版されたもので, OSの核となる部分であるプロセススケジューリングであったり, 大まかな設計自体は現代でも十分に通用するものです. しかし, ディスクの例がフロッピーディスクであったり, ページングが存在しないことなど, 今現在学習するには不適切な部分が存在します.
また, 自分がこの本で学習していたときに, 自分だったらこうするのにといった不満を感じる場面が存在しました.

例えば, 変数や関数の命名です. UNIXの流儀なのは理解できるのですが, とにかく略称が多く, 理解がワンテンポ遅れることがありました.
それに, 古いコンパイラでもコンパイルできるように, Cのマクロが多用されているところもまた理解の妨げになりました. これは現代に必要のないものです.
これらの問題点を解決したプロジェクトがA9Nです.

Resea

日本人の怒田 晟也さんという方が開発しているマイクロカーネルOSです. また, その解説はPDFで公開されています.
これは自分のやりたいことに非常に近いです. 軽量で, 拡張性が高く, HALが実装されていることや, しっかりとしたドキュメントが存在していることから明白です.
ほぼ完璧と言ってもいいプロジェクトです.
しかしながら, 私の提案するプロジェクトは, 私と同年代の十代(中高生)を対象とするもので, Reseaよりも低年齢向けです. その点において差別化を図ることができていると考えています.

3. あなたが自分の貴重な時間を使ってこのプロジェクトを実現したい理由 **(**任意**)**
あなたが、他の誰かと比べて、このプロジェクトを進めるのに適していると思う理由を教えてください。たとえばあなただけの強みや、過去の経験などがあれば教えてください。

小学生時代, 低レイヤー技術に関心があった私はOSC(Open Source Conference) 2017 Tokyoに参加しました.
そこで二人の方に低レイヤーやパケットの面白さを教わり, いつか完全自作OSを作成したいと思うようになりました.
また, そこで出会った数多くの大人に, オープンソースの素晴らしさを教わりました.
だからこそ, その二人や大人たちのように, 技術のおもしろさを伝えられる人間でありたいとも考えるようになりました.
それが自分の一番の原動力となっています.

私は現在高校に入学しておらず, 高卒認定試験を受け大学を受験しようと考えています.
一般的な17歳は殆どが高校生ですが, 私は自分のやりたいことだけに打ち込むため, このように特殊な形態を取っています. つまり, プロジェクトに多くの時間を割けるというわけです.
これが一つ, 適していると思う理由です.
小学生時代の憧れを実現させ, 人生でやり遂げたことの一つにしたいのです.

4. このプロジェクトについて現在までに取り組んだこと **(**任意**)**

類似品の調査や、仮説検証をするためのアンケート、実験、プロトタイプの開発など、今までに取り組んだことがあれば書いてください。なにがどこまでできていて、どういったことがこれから難しそうかを詳しく書いてくれたら、面談でのやりとりがスムーズになります。

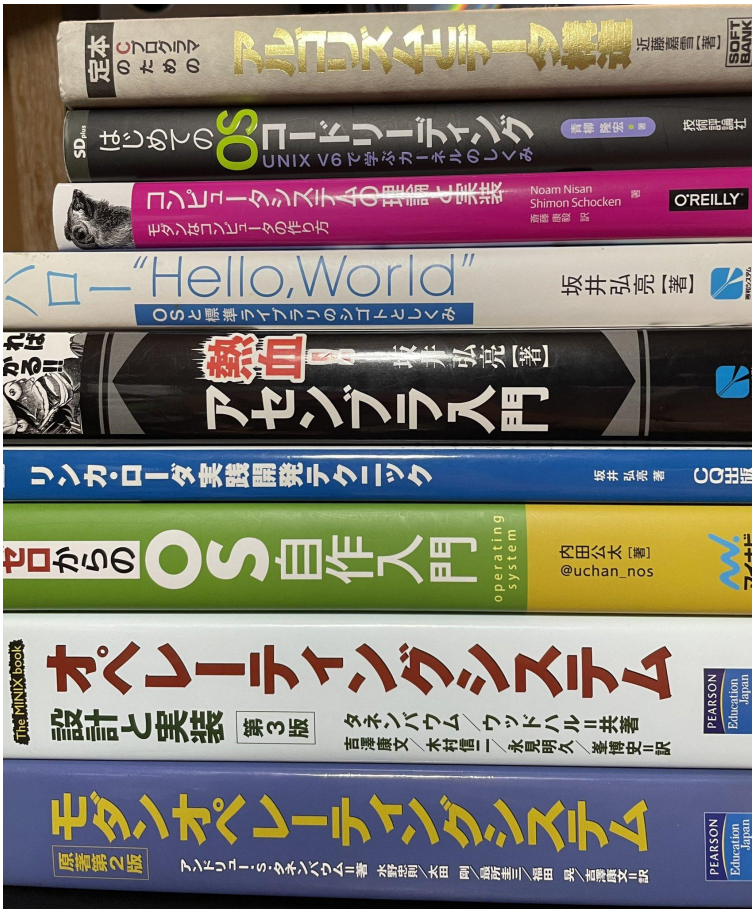
挫折

「完全自作OSを作る」という試みは2度挫折しています.
1度目はOSC2017の直後です. 自分には圧倒的に知識が足りていないことが分かりました. プロセスやMMU, 割り込みやデバイスドライバなど, OSがどのように作動して, どのような理論に裏付けられているかを知る方法すら分かりませんでした.
2度目は2年前, C++でOSを自作する本を読んだときです. なんとか写経はできて動作はできたものの, 完全に自作するには程遠く, やはり自分の知識不足を実感しました.
このままでは目標を達成できないと思い, 1年前から資料集めと学習を開始しました.

情報の収集

・書籍の収集

絶版になっている書籍が複数あり収集に時間がかかりましたが, 以下の書籍を入手し学習しました. (一部, 元々所持していたものもあります)



・モダンオペレーティングシステム
・オペレーティングシステム: 設計と実装
・ゼロからのOS自作入門
・リンカ・ローダ実践開発テクニック
・熱血！アセンブラ入門
・ハロー”Hello, World”
・コンピュータシステムの理論と実装
・はじめてのOSコードリーディング
・アルゴリズムとデータ構造

プロセススケジューラのプロトタイプを書いてみたりはしましたが, 基本的にはこれからの実装を考えています．そのための知識を付けてきました.

5. 提案者がこれまで制作したソフトウェアまたはハードウェア

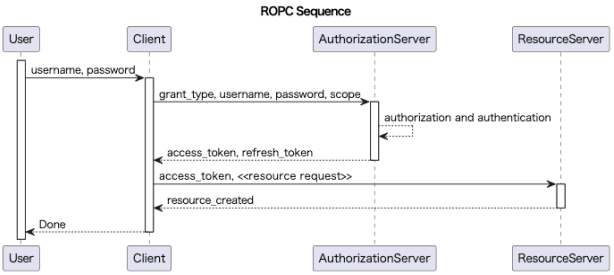
- 自分が、このプロジェクトを進めるにあたり十分な能力があることを、アピールしてください。（特に 4.でまだプロトタイプなどの開発をやっていないと解答した方は、ご自身の能力を強くアピールしてください。）
- これまでの活動実績が載っているホームページや、GitHubのアカウント、YouTubeチャネル等がある場合も、こちらでアピールしてください。
- フォーマットは自由です。図表や画像も使用できます。ページ数も制限はありません。複数人で開発した場合は、どの部分を担当したのか、明確に記述してください。

ソフトウェア

Githubを自分ではあまり使ってこなかったので, 量は少ないですが提示させていただきます.

・hss-Python (https://github.com/horizon2038/hss-python)

DDD(Domain-Driven Design, ドメイン駆動設計)を元に作成した, RESTfulなOAuth2の認証+認可サーバーです(OAuthは認可のプロトコルであり認証のものではないが, 例外として認証も含んだResource Owner Password Credentials Grantを使用している).
Python3で書かれています.

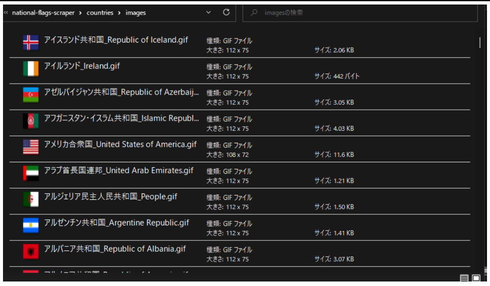


左: ROPCのフロー 右: 使用したアーキテクチャ

今となっては書き直したいところが多々出てきています. 特にOAuth2.1ではROPCは非推奨なので, その内Authorizatiojn Code Grantに書き直したいところです.

・National-Flags-Scraper

(https://github.com/horizon2038/national-flags-scraper)
外務省のページから国名と国旗をスクレイピングするシステムです.
弟が国旗に関心があるので, 国旗ゲームの素材を得るために作成しました.



左: 得られた画像 右: 取得した国名とURLを纏めたjson

・HyperCasualSystem (プログラミングスクールで発表した資料)

Unity用のハイパーカジュアルゲームテンプレートです.
これは, DDDとイベント駆動型アーキテクチャにより作成したものです.

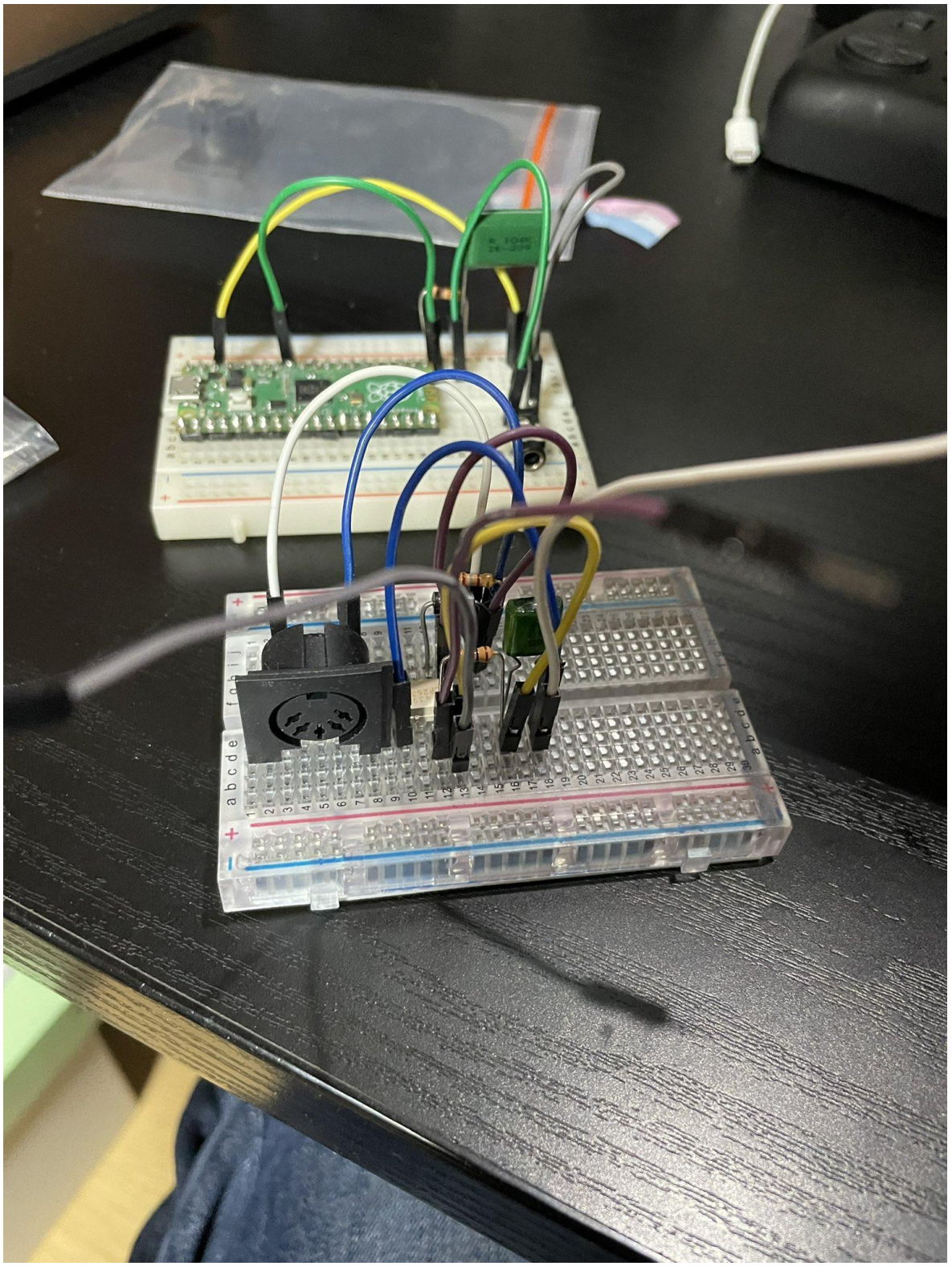
・NES-ROM-Analyzer

```
PS D:\Development\Program\hNE> ./rom_analyzer sample1.nes
nes: NES□□□□
program_rom_size: 32768B
charactor_rom_size: 8192B
======PROGRAM SECTION======
78   A2   FF   9A   A9   00   8D   00   20   8D   01   20   A9   3F   8D   06
20   A9   00   8D   06   20   A2   00   A0   10   BD   51   80   8D   07   20
E8   88   D0   F6   A9   21   8D   06   20   A9   C9   8D   06   20   A2   00
A0   0D   BD   61   80   8D   07   20   E8   88   D0   F6   A9   00   8D   05
20   8D   05   20   A9   08   8D   00   20   A9   1E   8D   01   20   4C   4E
80   0F   00   10   20   0F   06   16   26   0F   08   18   28   0F   0A   1A
2A   48   45   4C   4C   4F   2C   20   57   4F   52   4C   44   21   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
00   00   00   00   00   00   00   00   00   00   00   00   00   00   00   00
```

ファミリーコンピューター(NES)のROMを解析し, プログラムROMとキャラクターROMのサイズ表示・パースを行うものです.

ハードウェア

MIDI Receiver



Raspberry Pi Pico用のMIDI受信機構です. また, 後ろに写っているのはRaspberry Pi PicoのPWMで音を再生するモジュールです.

6. 週あたりの作業時間の目安

作業時間*:* 学期中 ×時間 夏休み中 ◇時間
作業時間: 21-35時間 (1日3-5時間)

7. 開発費の使用計画

開発に関わる費用が合計 *50* 万円まで補助されます。現時点で未定の項目があっても構いません。支出項目については採択後*PM*との相談により決定します。 例*:* ・××デバイスの購入 *8,000* 円、◇◇ 機能の実装のため ・△△ソフトウエア *12,000* 円、〇〇 を作成するため ・☆☆デジタルデータ *20,000* 円、□□で使用するため ・使途未定（開発で必要が生じた場合の予備）*: 240,000*円
・開発用ノートPC: 300,000円 OSの開発用 ・用途未定: 200, 000円 予備として. また, 異なるアーキテクチャへの移植用としてRISC-Vボードもしくは Raspberry Pi購入の可能性あり.

8. 自己アピール

その他、まだアピールしきれていない、得意なことや、ほかの人にはないような経験があればアピールしてください。必ずしも提案内容と関連している必要はありません。

基本情報

My Twitter
私は17歳(2005年生まれ)です.
プログラム関係以外では, 作曲(電子音楽), 考古学(古代エジプト), 美術, 3DCG/VFX, デザインに関心があります.
Twitterでは制作物や進捗をアップロードしたり, 雑多なことを呟いています.

プログラム

低レイヤーからソフトウェアアーキテクチャ, ネットワークやゲーム制作等に関心があり, 学習しています.
小学校4年生のとき, Raspberry Piを手に入れたことがキッカケでプログラムの道へと入り, その少し後からiOSアプリ開発(Objective-C + Swift), Tensorflow+OpenCVでの顔認識, DiscordのBot開発, ゲーム開発と続き, 今へ至ります.
基本的にはC/C++とC#, Python, Swiftを書きます.
小学生のときに参加したMaker FaireとOSCで人生が変わりました. それらのイベントには感謝してもしきれません.

作曲

主に電子音楽を作曲していて, EDM(Future House, Future Bass, Progressive House)やIDMを制作しています. 制作物はTwitterにアップロードしたりしています.
また最近は和声学や対位法, 管弦楽法にも手を出しています.

考古学

小学校2年生のときに古代エジプトが好きになり, ヒエログリフや時代背景を学んでいました. なので, 少しヒエログリフを読むことができます. これはローマ字に換算したよくあるエセヒエログリフではなく, 実際の壁画やステラなどです. デモティックは読めません.

美術

絵を描くことが好きで, パースや美術解剖学を学んでいます.

3DCG/VFX

実写合成やモデリングをやっています. 3年前には, CPUメーカーであるAMDのVlogチャレンジでちょっとしたモノをもらったことがあります.
実際の動画

以上のようにかなり趣味の範囲が広いですが, すべてをマスターして器用裕福になりたいと考えています.