* 目次
  + はじめに
  + 用意するもの
  + PlatformIOの導入
  + プログラムの取得
  + プログラムの実行
  + ライブラリ
  + ファイルとフォルダの構成
  + 共同開発の環境構築
  + 他の人によるプログラムの変更を取得する
  + 自分によるプログラムの変更を他の人に反映させる
  + 最後に
* はじめに

Arduino IDEだと、タイプミスをしてもエラー表示が出なかったり、タイピング中に候補が出てこなかったりする。PlatformIOと比べて取っつきやすい反面、使いにくい点が多い。そこで、PlatformIOを使おうと考えた。

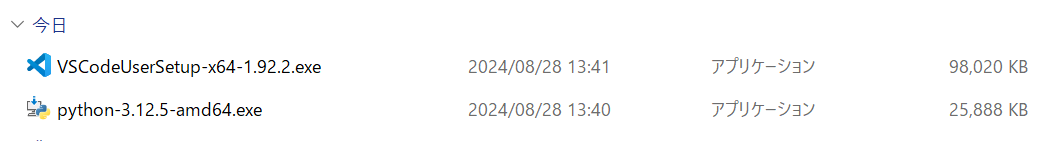
僕がやろうと言い出した分、他のメンバーに使い方を説明する責任が僕自身に生じる。この資料では、PlatformIOを使ったマイコンのプログラミングの仕方を説明する。この資料を見てわからないところがあったり、この資料を読むのが面倒だと感じたりする場合などは、遠慮なく僕に相談してほしい。

* 用意するもの
  + パソコン
  + マイコン
  + マイコンとパソコンを繋げるケーブル
  + やる気
  + 気合
  + 根性
* PlatformIOの導入

[このサイト](https://logikara.blog/install_platform/)を参考にしてほしい。Visual Studio CodeとPythonをインストールし、Visual Studio Codeの拡張機能としてPlatformIOをインストールする必要があるとのこと。念のため、ここでも軽く説明する。

まず、インストーラーをダウンロードする必要がある。インストーラーは、[ここ](https://code.visualstudio.com/)と[ここ](https://www.python.org/downloads/)にアクセスして、ダウンロードできそうなボタンを適当に押せば、ダウンロードすることができる。これとか、これのことである。

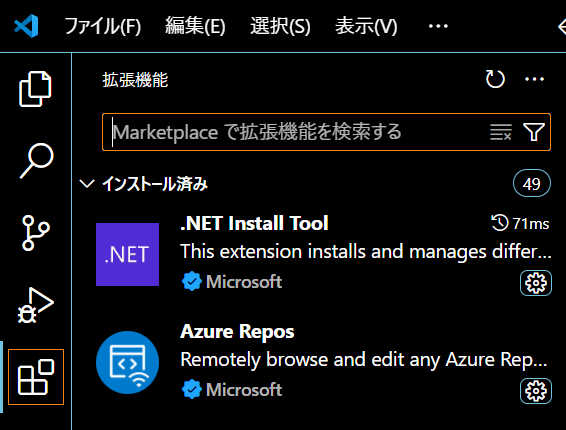
次に、Windows上のこんな感じのアイコンをクリックして、エクスプローラーと呼ばれる画面を表示させ、をクリックし、ダウンロードフォルダを開く。



そうすると、先ほどダウンロードしたインストーラー２つが表示されるはずなので、それぞれクリックしてインストールを開始する。インストーラーから何か訊かれると思う。Pythonのインストーラーの場合、たら、適当に肯定的な返答をしたら問題ない（はず）。

インストールが終わると、パソコンのどこか（例えばインストールされたアプリを検索する画面など）にVisual Studio Codeのアイコンがあるはずなので、それをクリックして起動させてほしい。Visual Studio Codeインストール直後の初めての起動になるので、何かしらの設定を要求されるかもしれないが、それは適当にやってほしい。

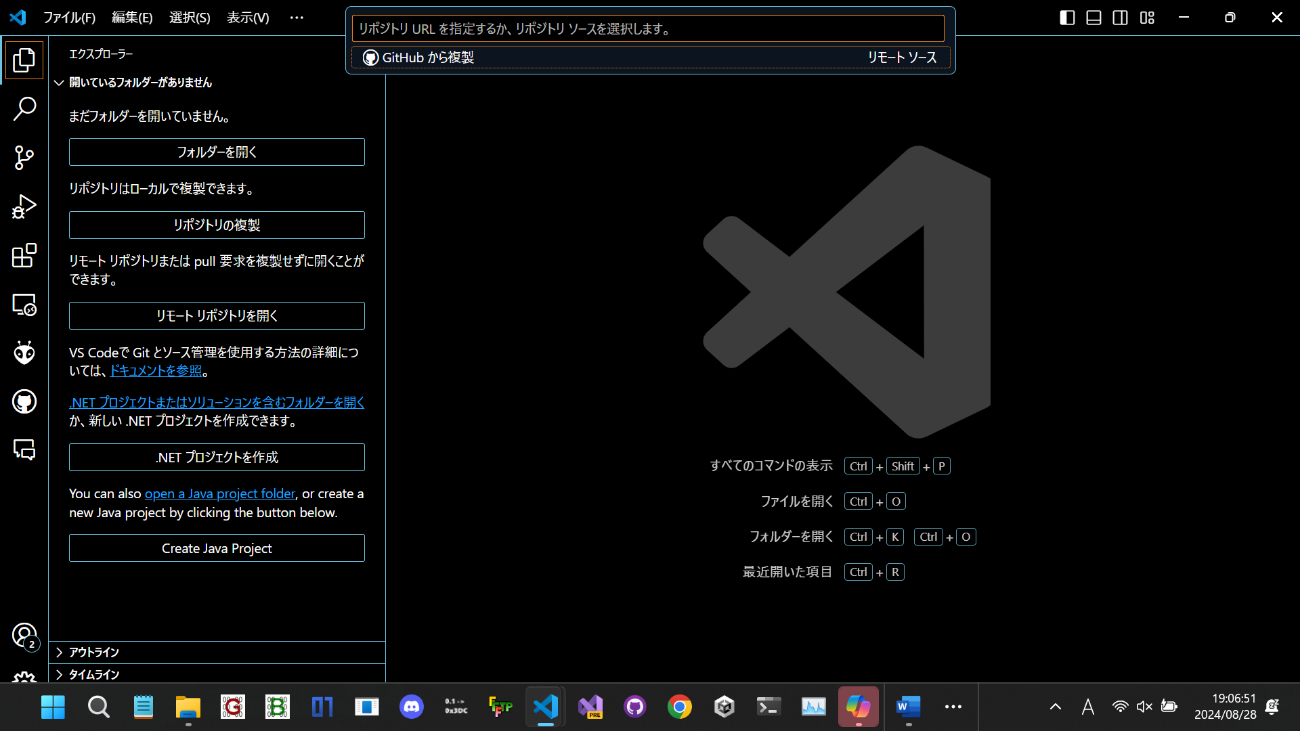
Visual Studio Codeの左端にあるをクリックすると、拡張機能を検索する画面が出てくる。ここで、『platformio』と検索すると、蟻🐜と宇宙人👽を合成したような見た目をした生命体のロゴが出現する。それをクリックすると、インストールという選択肢が見えるはずである。その選択肢を押してほしい。ちなみに、これと同じように『japan』と検索しインストールすれば、Visual Studio Codeを日本語にすることができる。

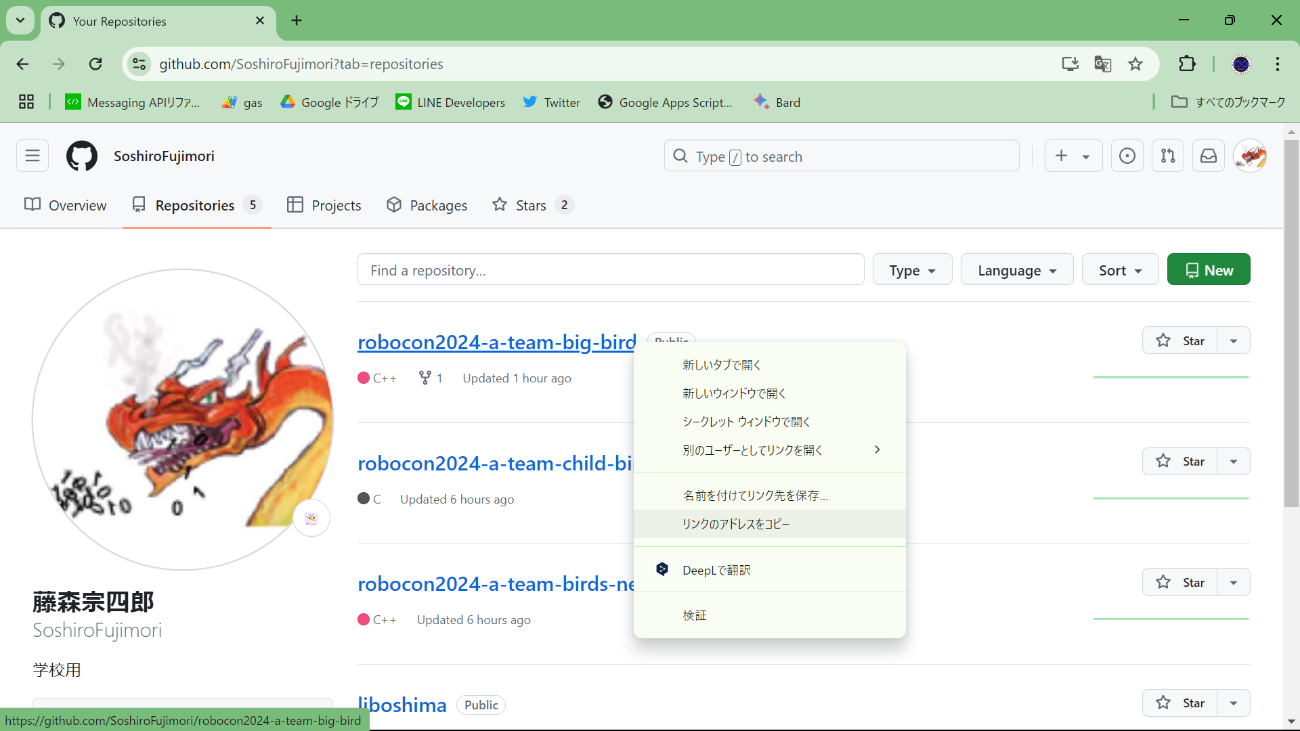
後はいい感じに待てば、PlatformIOの導入は完了である。これでプログラムをする環境は整った。

* プログラムの取得

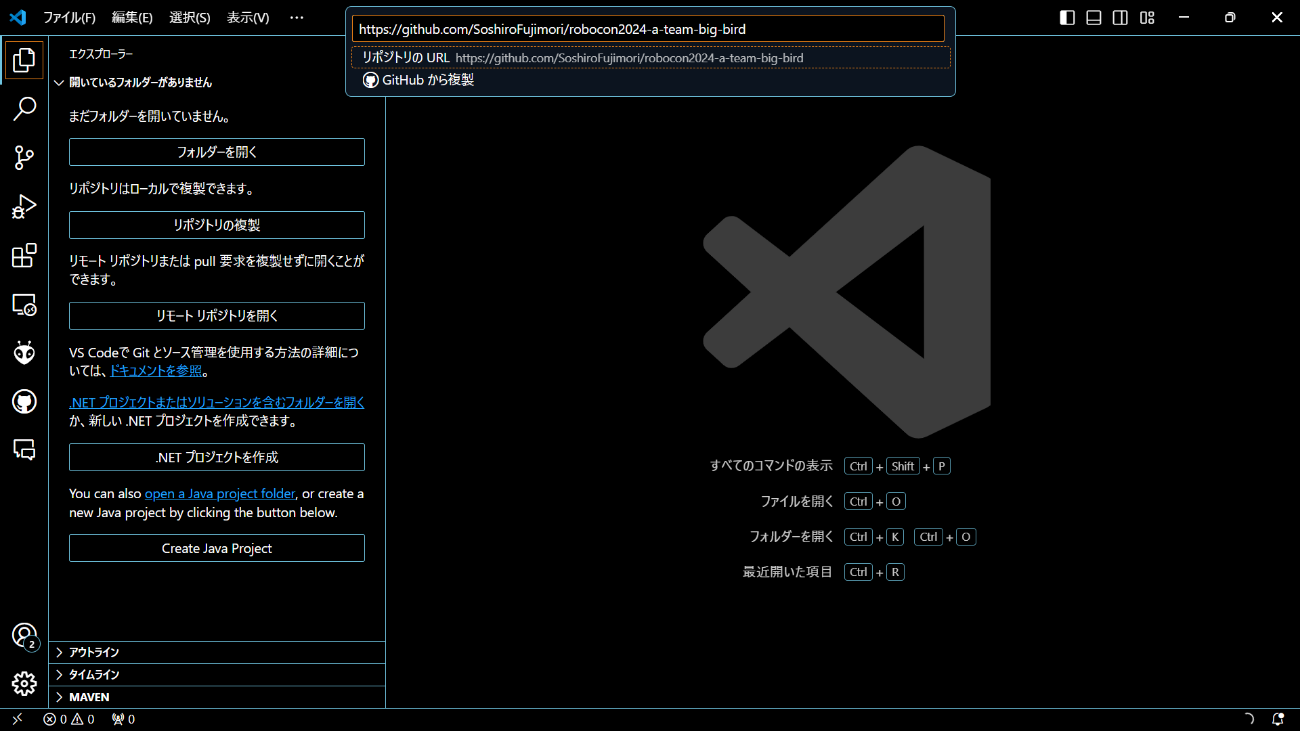
　Visual Studioを起動させ、左上にあるが選択されていることを確認する。次にをクリックし、テキスト入力ができる画面を表示させる。



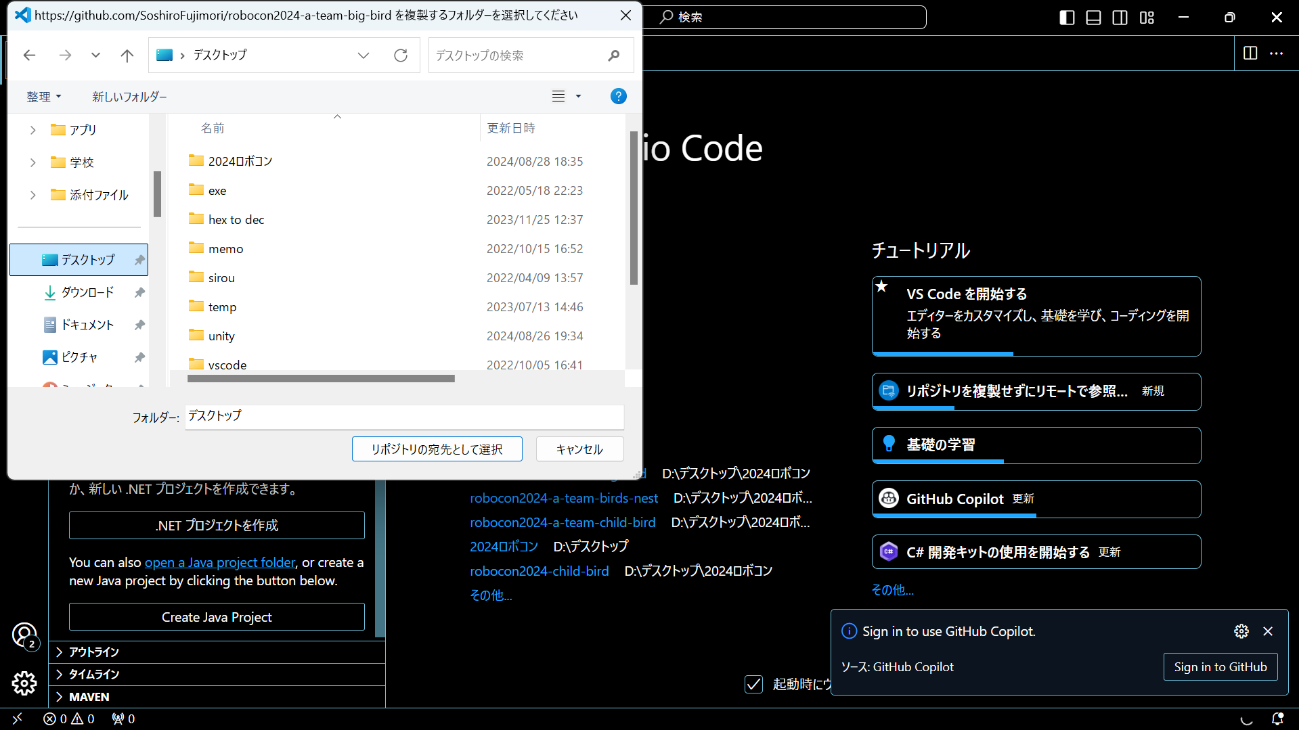
ここに、プログラミングをしたいロボットの『プログラムのURL』を貼り付けてほしい。そのURLは[このサイト](https://github.com/SoshiroFujimori?tab=repositories)から手に入れることができる。例えば、もしAチームの親鳥のプログラミングをしたい場合は、にカーソルを合わせ、右クリックをし、を選択することでプログラムのURLを取得することができる。



　URLを貼り付けるとこのような画面になるのでを選択する。



選択すると、プログラムの保存場所を訊かれる。自分が分かる適当な場所を選んでほしい。ここではを選択した。で画面を閉じる。



と訊かれるので、を選ぶ。



さらにこのような画面になるので、を選択する。

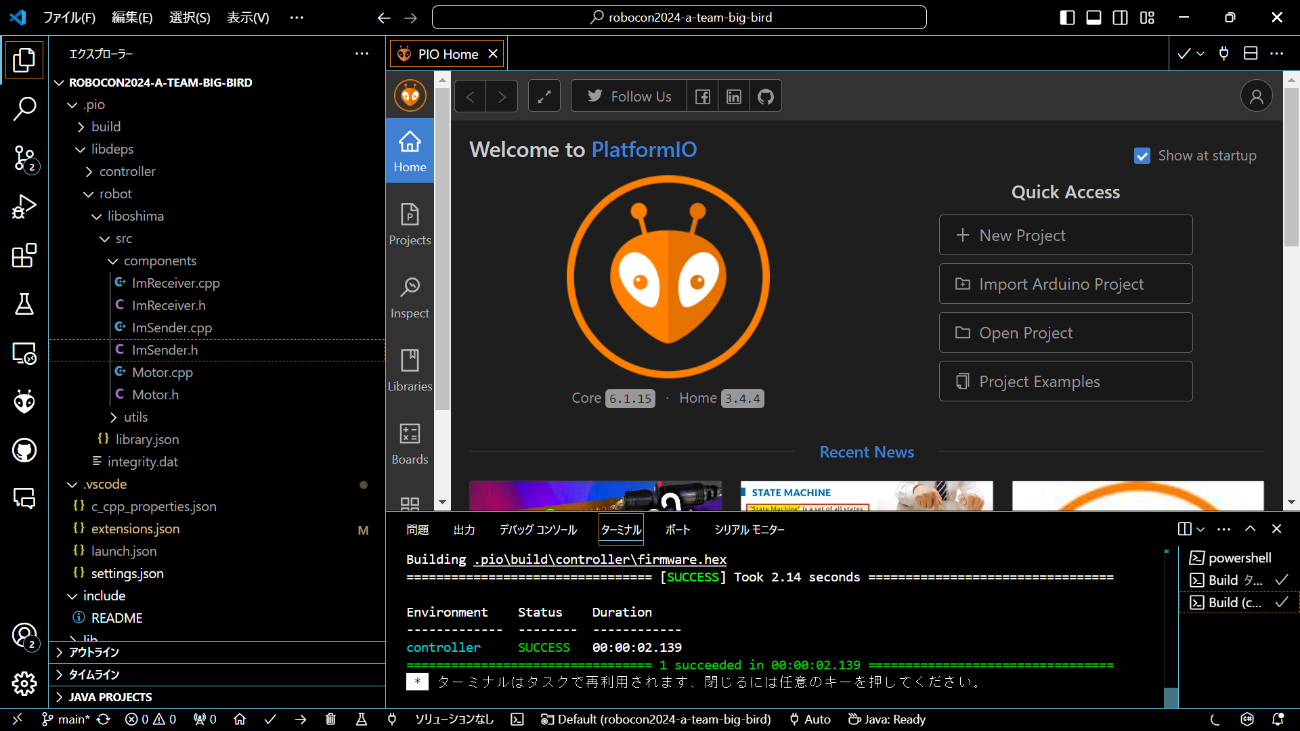


　これで、プログラムの取得は完了である。

* プログラムの実行

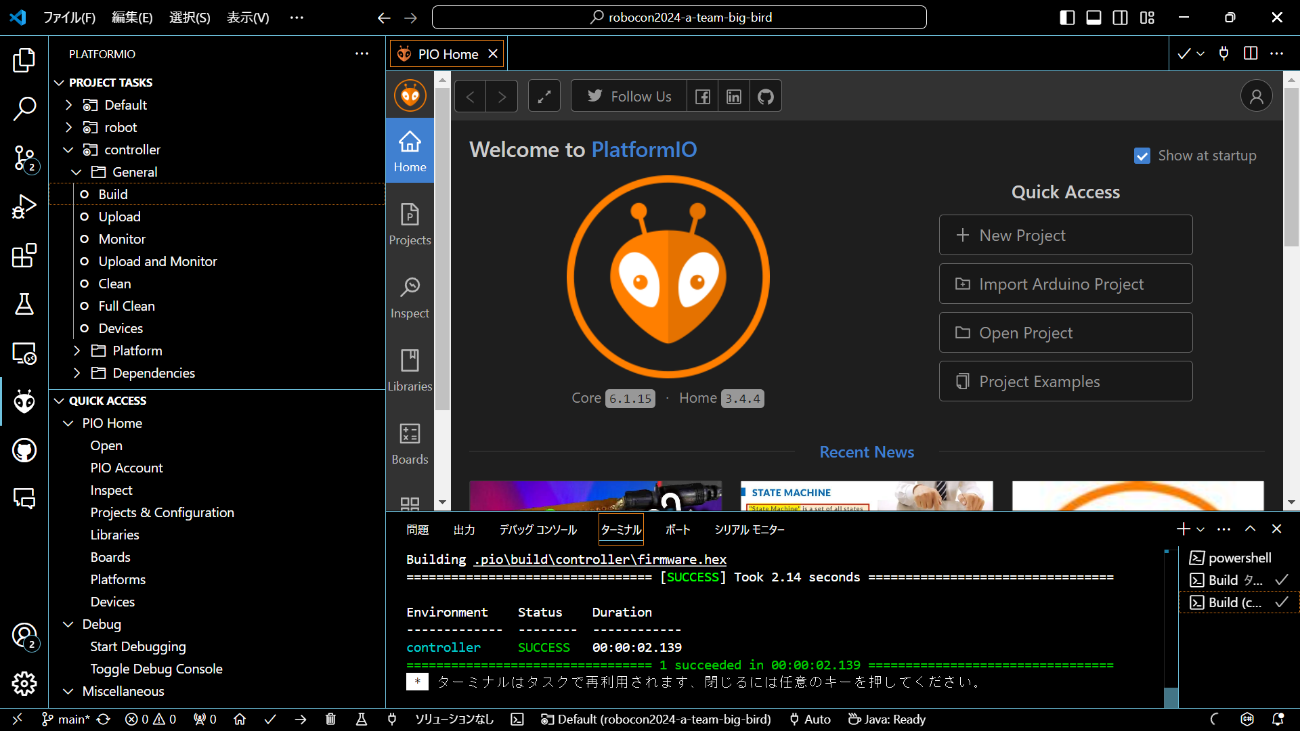
1. マイコンをパソコンにつなげる。
2. （ロボット本体のプログラムをマイコンにアップロードする場合）

右上のを押す。



1. （コントローラーのプログラムをマイコンにアップロードする場合）

左中央にある ->  ->  ->  -> の順でクリックする。



5

4

3

2

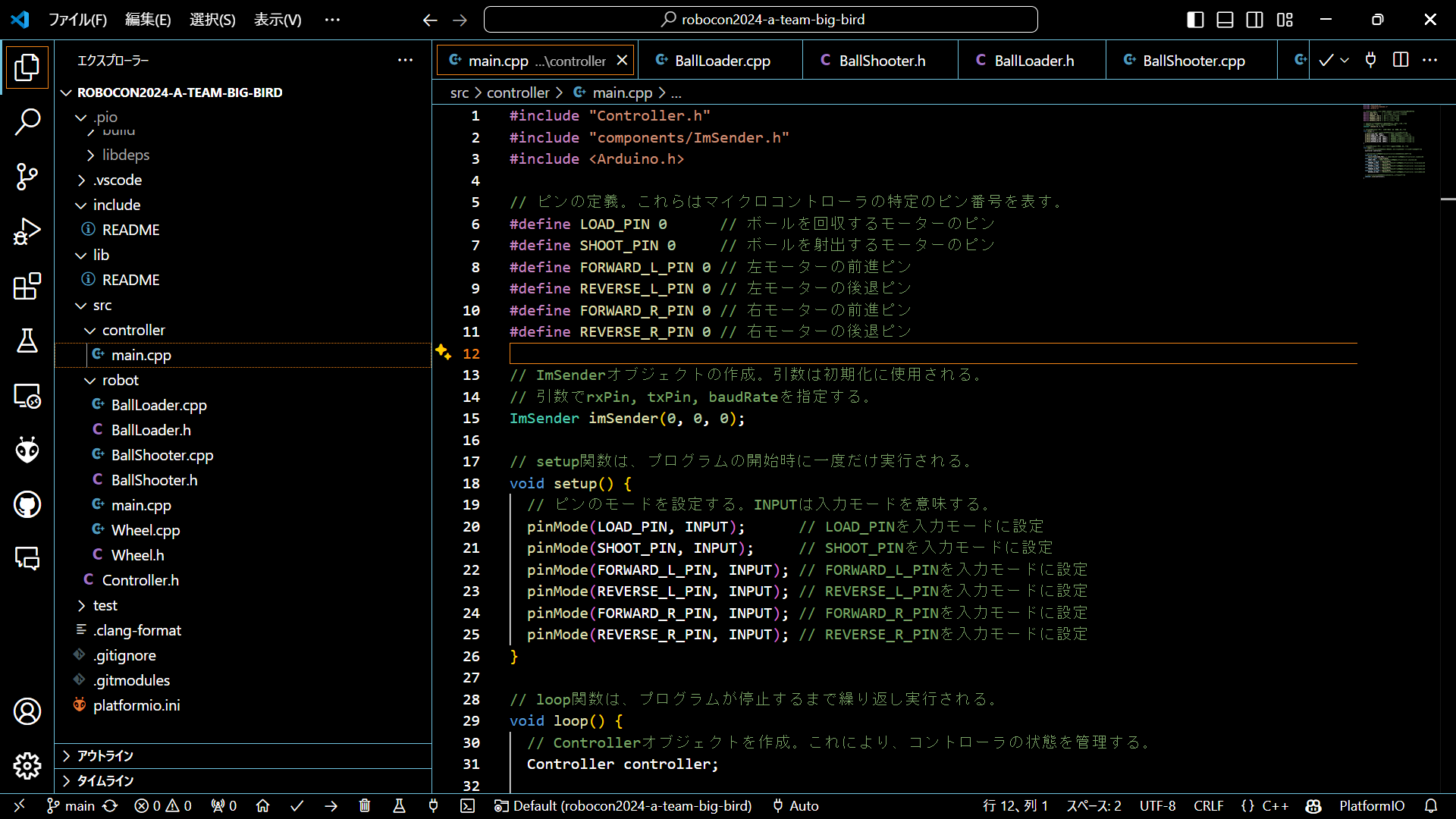
1

* ライブラリ

コントローラーのプログラミングをしたい場合はsrc/controller/main.cppを、ロボット本体のプログラミングをしたい場合はsrc/robot/main.cppを開く。ここでは、後者を見ていく。

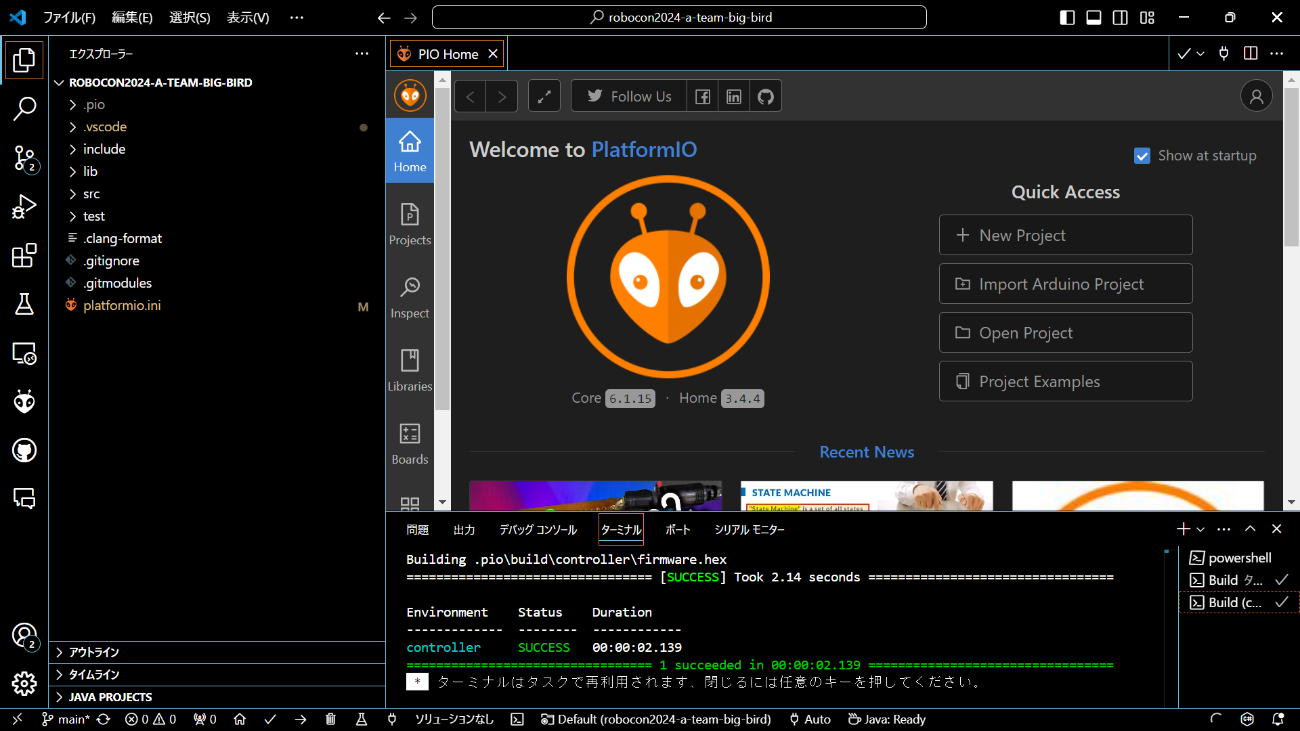
プログラム先頭にとあると思う。（ない場合は必要に応じて追加してください。）ここで、ライブラリを使用することを宣言している。自分で実装していないのに、使えるものはすべてライブラリから来ている。例えば、はpinModeという処理を呼び出しているが、このような処理は自分自身で実装していない。（void pinMode(){}というようなコードはこのmain.cppファイルには見当たらない。）では、どこから来たのか？ライブラリから来たのである（…たぶん）。をすることで、pinModeという処理を自分で実装せずとも使えるようにしている。

はimSenderという名前でimの受信機を作成している。()内にはピン番号２つとボーレートを順に指定する必要があり、ここでは適当に0にしている。ImSenderという型（int『型』とかfloat『型』とかのあの型）を使用しているが、これはをしているため使えるようになっている。これはliboshimaというライブラリの一部で、[ここから](https://github.com/SoshiroFujimori/liboshima)取り入れたものである。



* フォルダとファイルの構成

　左上にあるを押すと、このような画面になる。左側にプログラムを構成するすべてのファイルやフォルダが載っている。これらのなかから、特に重要なものからいくつか説明する。



* + platformio.ini

プロジェクトの設定ファイル。使用するプラットフォーム、ボード、フレームワーク、ビルドオプション、ライブラリの依存関係などを定義する。ちなみに、このファイルはPlatformIOのUIを使って編集することもできる。

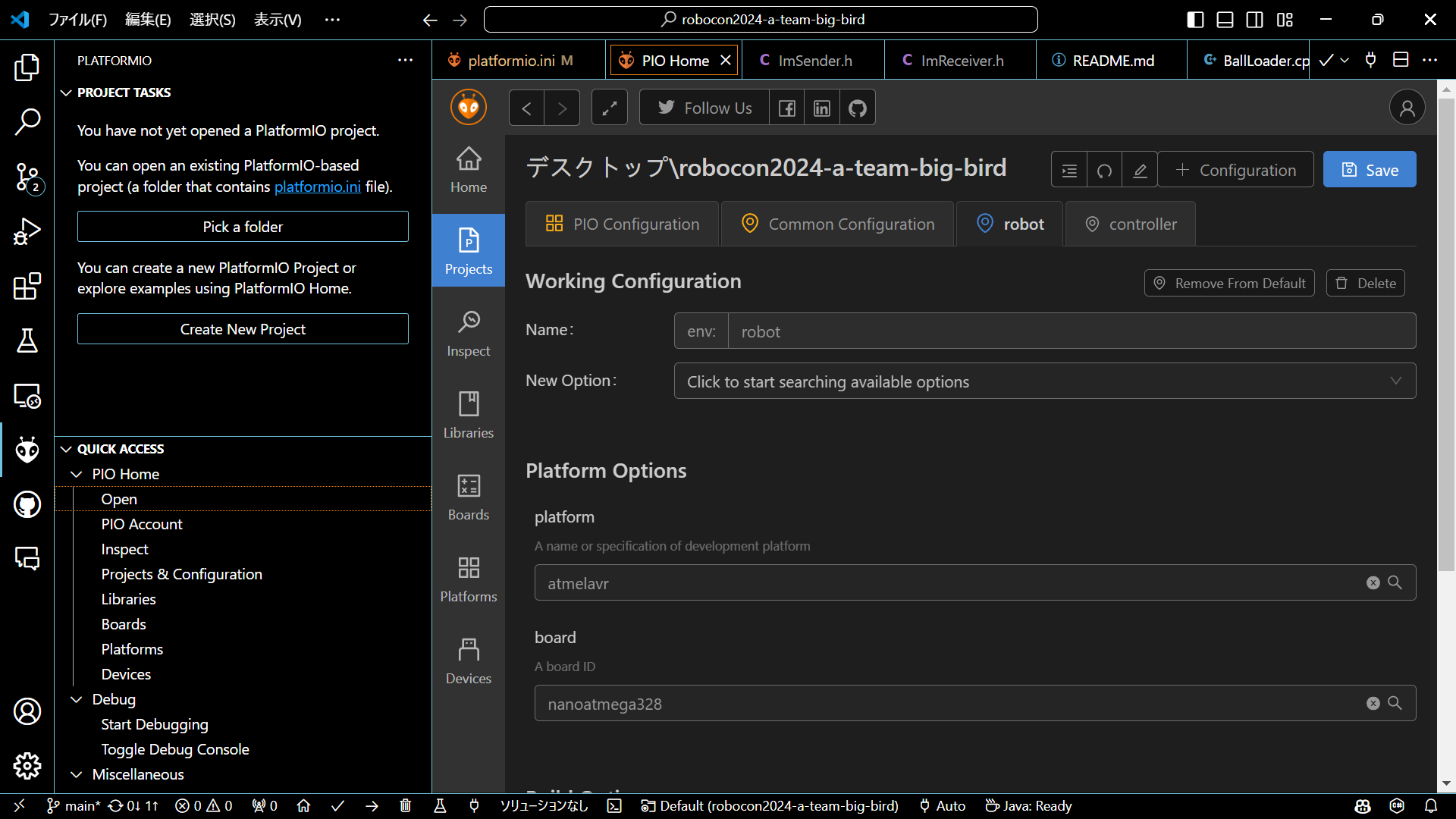
テキスト

自動的に生成された説明

[PlatformIOレジストリ](https://registry.platformio.org/libraries/soshirofujimori/liboshima)からver.0.0.6のliboshimaライブラリを取得。また、PlatformIOのシリアルモニターのボーレートを設定している。このライブラリは[env]で示されているため、すべてのマイコンで利用可能になる。ちなみに、PlatformIOレジストリのプログラムは[Githubリポジトリ](https://github.com/SoshiroFujimori/liboshima)から来ている。

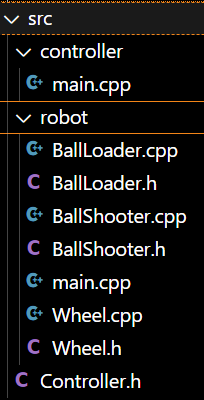
デフォルトのマイコンとして『robot』（下の行で定義済み）を設定。ビルドをする際に特定のマイコンを指定しない限り、robotが使用される。

2つの異なるマイコン（robotとcontroller）を定義。それぞれに異なるソースコードフォルダをビルド対象として指定したり、マイコンの種類を指定したりしている。



* + src/

プロジェクトのソースコードを格納するフォルダ。ここにあるファイルがビルド対象になる。より正確に述べると、controllerのマイコンとしてビルドした場合はcontrollerフォルダ内がビルドされ、robotのマイコンとしてビルドした場合はrobotフォルダ内がビルドさる。



* + lib/

手動で管理するライブラリを格納するフォルダ。コピー＆ペーストで扱うライブラリがここに入る（？）。今回のロボット制作では恐らく使わない。ちなみに、READMEファイルにこのフォルダの使い方が記載されている。



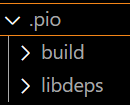
* + include/

ヘッダファイルを格納するフォルダ。本来であればsrc/にあるヘッダファイルはこのフォルダに置かないといけない（？）。今回のロボット制作では恐らく使わない。こちらも同様にREADMEファイルにこのフォルダの使い方が記載されている。



* + .pio/

ビルドによって生じた実行ファイル（.hexや.elfなど）や、platformio.iniのlib\_depsによってダウンロードされたライブラリなどを格納するフォルダ。（このディレクトリは通常、Gitなどのバージョン管理システムでは無視される。）



* + .gitignore

Gitを使用する場合に、バージョン管理から除外するファイルやフォルダを指定するファイル。

テキスト

自動的に生成された説明

* + .clang-format

コードのフォーマットを統一するために使用される設定ファイル。ここでは、LLVMスタイルに従うように設定している。

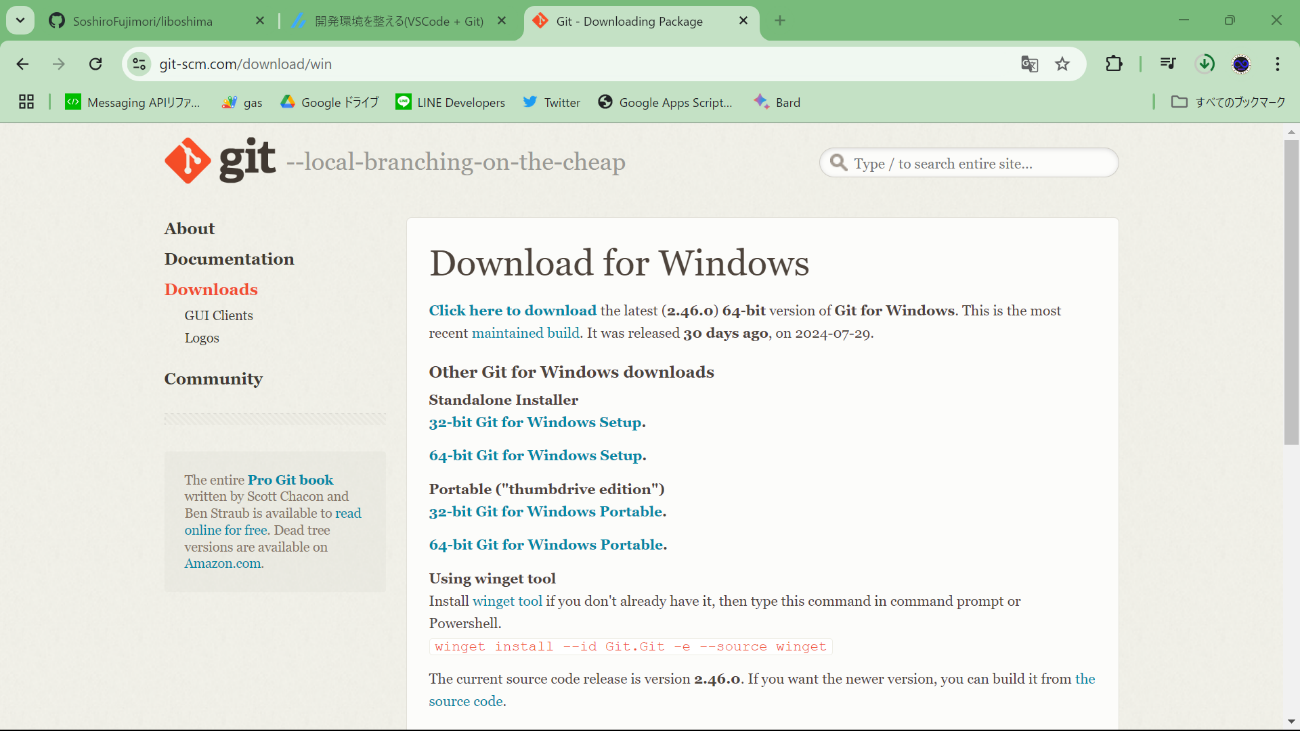
グラフィカル ユーザー インターフェイス

中程度の精度で自動的に生成された説明

* 共同開発の環境構築

[このサイト](https://zenn.dev/ojk/books/github-vscode/viewer/install)が参考になる。まず、Gitというソフトをインストールする必要がある。

[このサイト](https://git-scm.com/download/win)のをクリックしインストーラーをダウンロードする。ダウンロードしたインストーラーをエクスプローラーのダウンロードフォルダから見つけ、実行する。インストーラーから何か訊かれたら、適当に答えるか僕に訊いてほしい。



　次に、Githubアカウントを作成する必要がある。[ここ](https://github.com/signup)にアクセスし、メールアドレス、パスワードなどを適当に設定する。作成したアカウント名は僕に教えてほしい。（リポジトリの管理権限を与えるために使用する。）

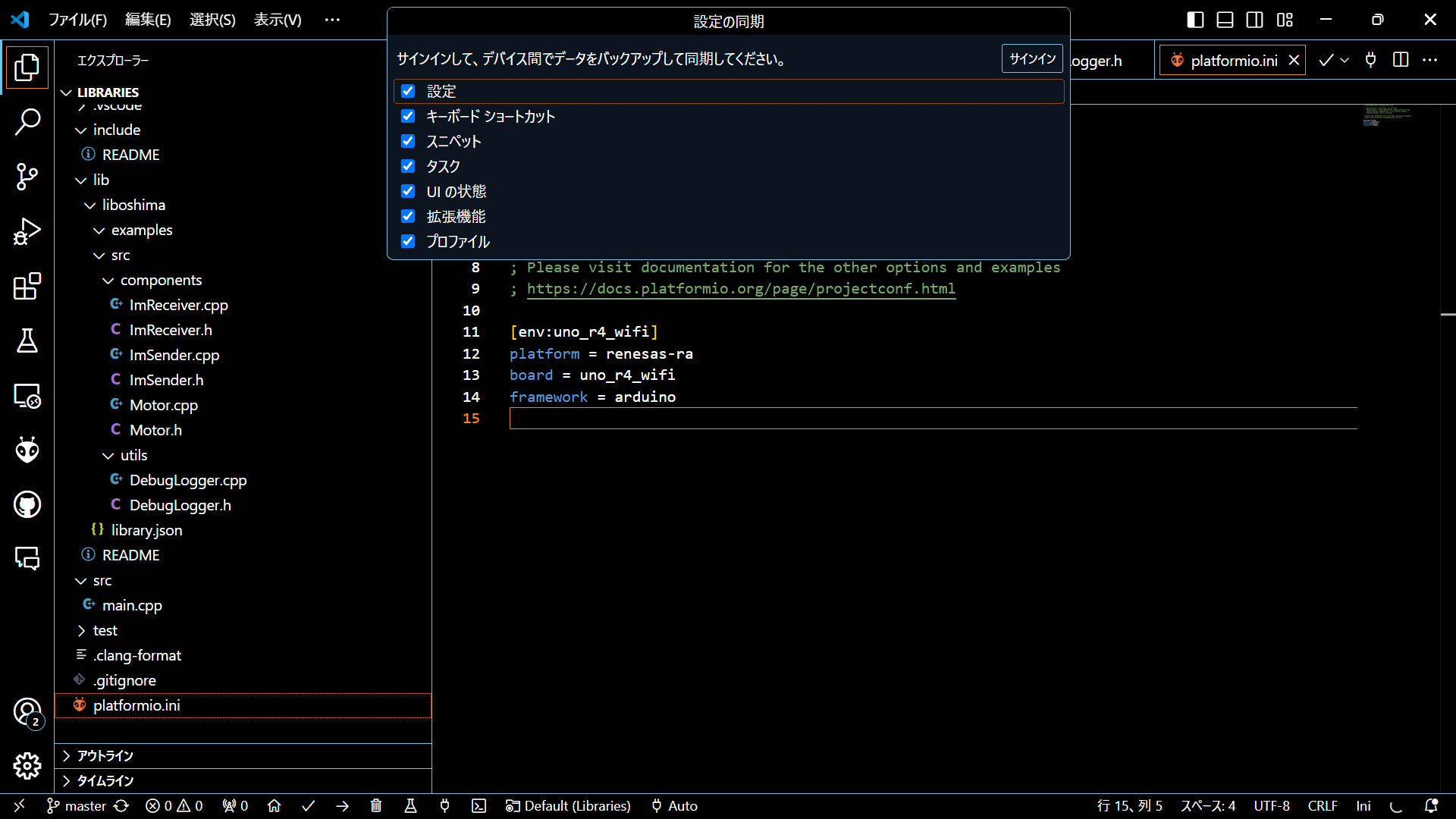
　最後に、Visual Studio CodeとGithubアカウントを紐づける。Visual Studio Codeを起動させ、左下にある挿絵 が含まれている画像

自動的に生成された説明をクリックし、を選択する。

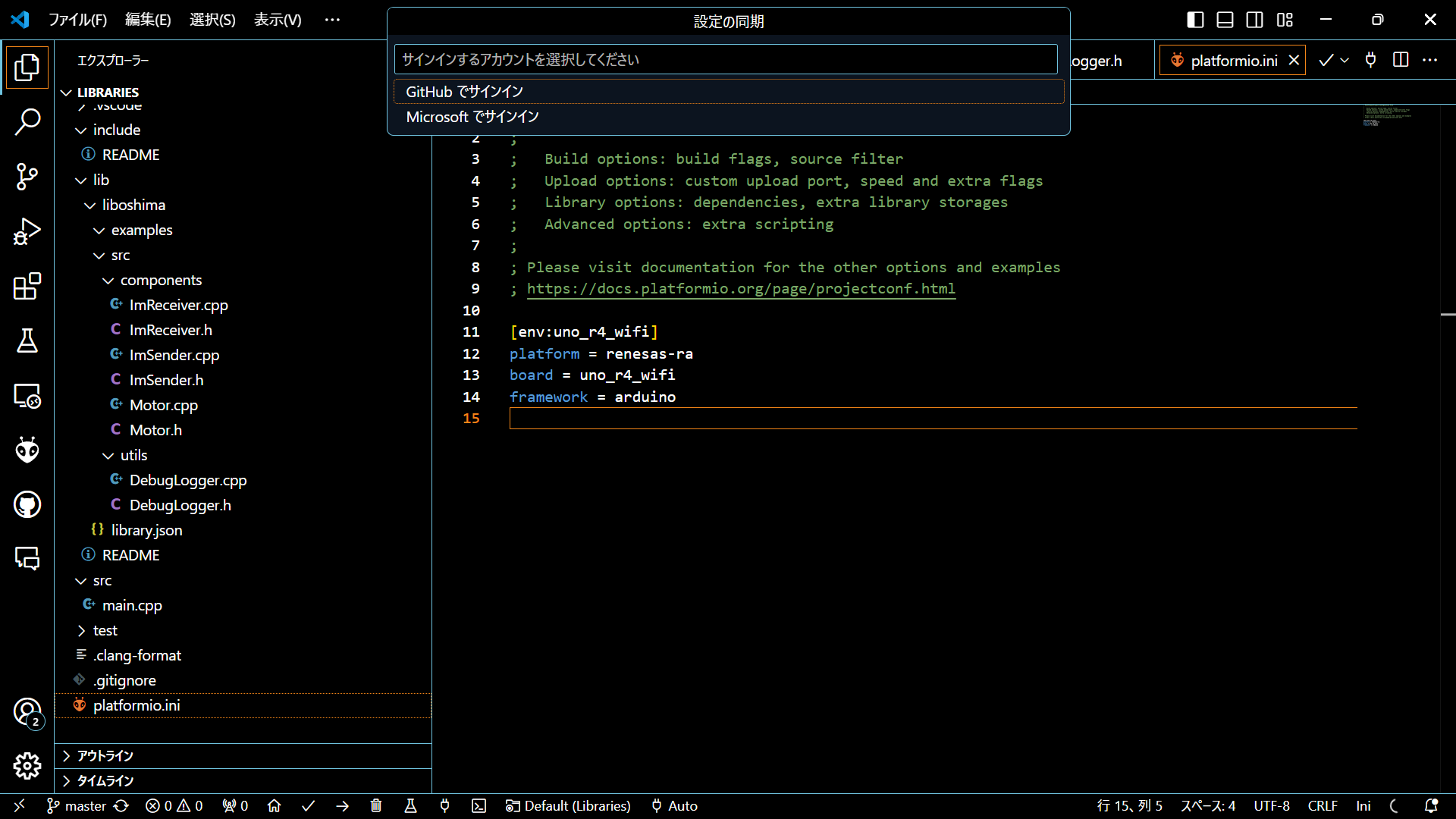
コンピューターの画面のスクリーンショット

自動的に生成された説明

ここで、画面上側にというボタンが出てくるのでそこをクリックする。



を選択する。すると、Google Chromeが立ち上がり、ログインするためのサイトが表示される。適当に進める。



Visual Studio Codeの画面左下にあるアイコン

自動的に生成された説明をクリックし、ログインしたGithubアカウントが表示されれば、ログイン成功である。

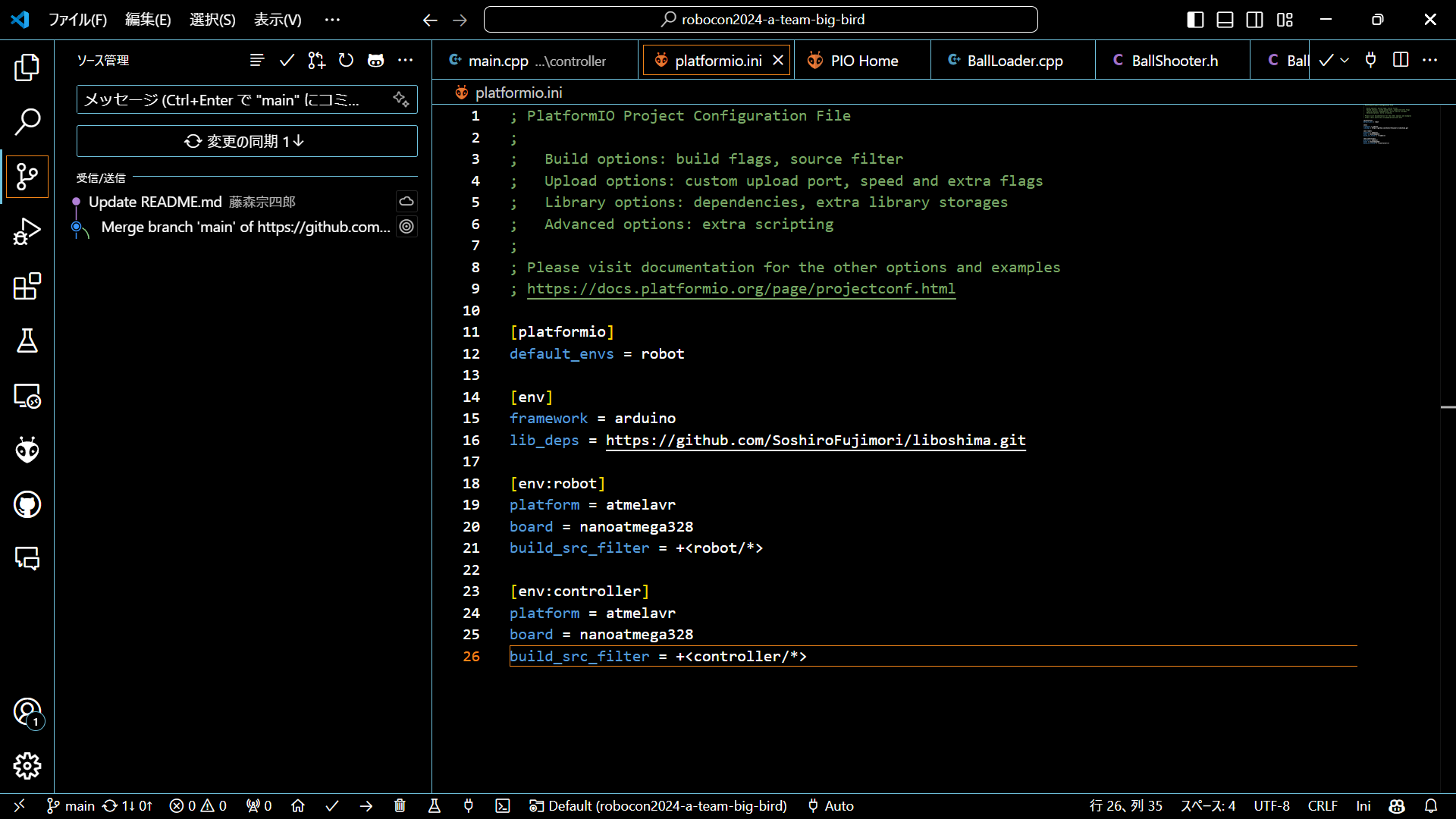
テキスト

自動的に生成された説明

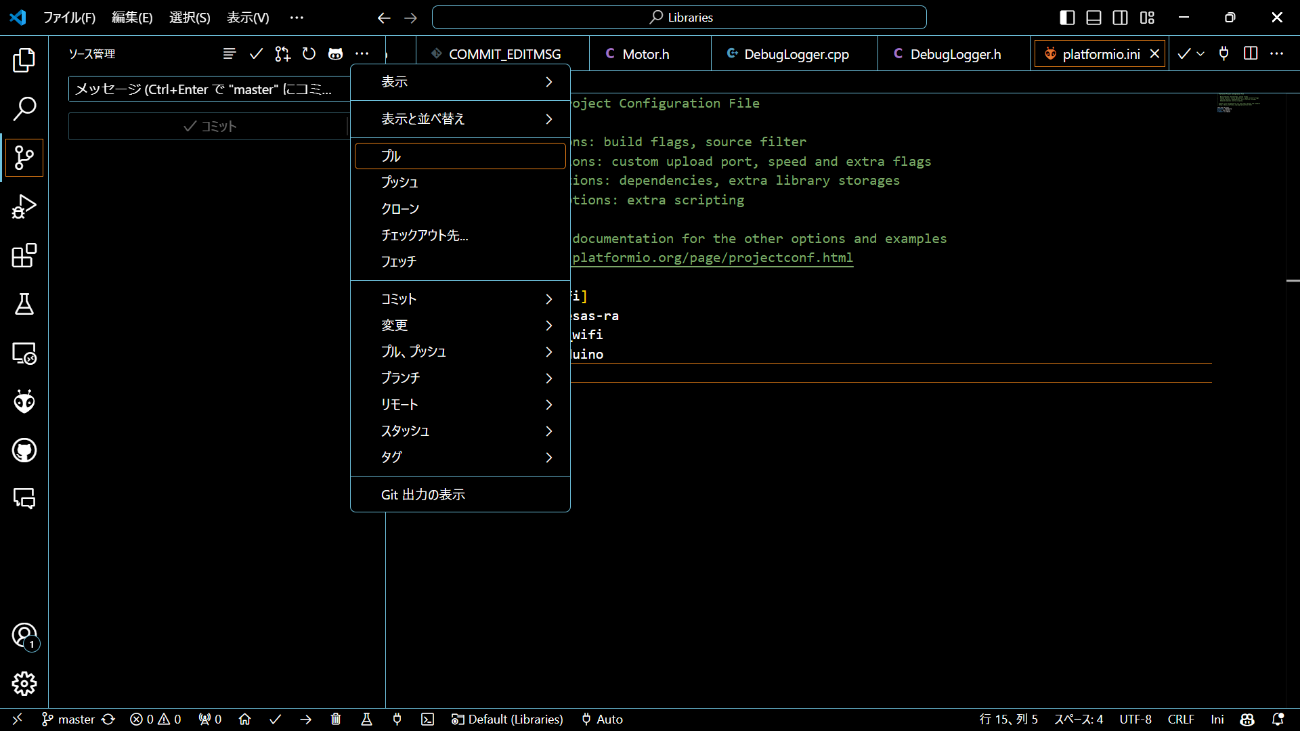
* 他の人によるプログラムの変更を取得する

[このサイト](https://zenn.dev/ojk/books/github-vscode/viewer/pull-push)が参考になる。Visual Studio Codeの画面左上にあるアイコン

自動的に生成された説明をクリックする。ここでをクリックすることで他の人によるプログラムの変更を取得することができる。



もしの表示が無ければ、からを選んで取得する。



もしエラーが出たら教えて欲しい。

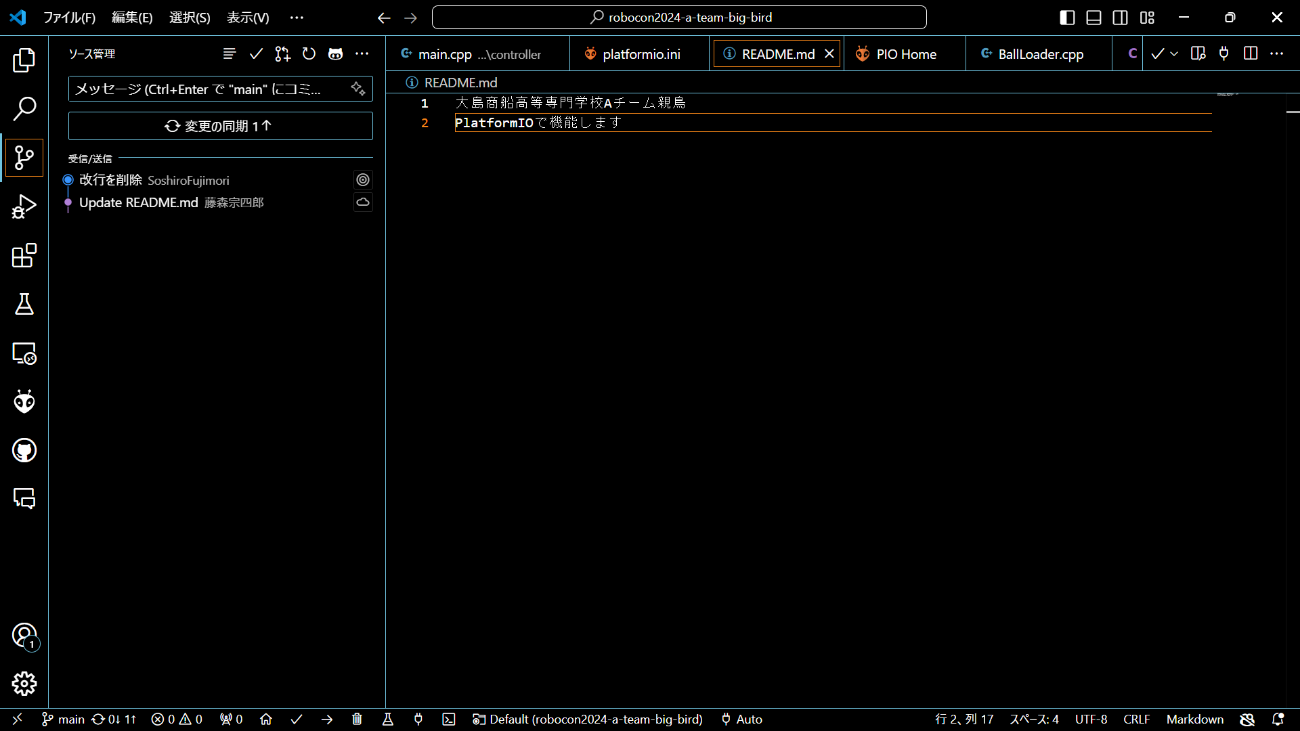
* 自分によるプログラムの変更を他の人に反映させる

プログラムに対しどのような変更を加えたのかを、に打ち込む。その状態で、をクリックし、出てくる画面にと答える。

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

最後にを押して、完了である。



もしエラーが出たら教えてほしい。

* 最後に

思いのほか説明が長くなってしまった。再び述べることになってしまうが、もし読むのが面倒だと感じた場合はご遠慮なく質問してほしい。

また、誤植やわかりにくいところ等あれば教えてほしい。連絡先を載せておく。

メール：[i22110@oshima.kosen-ac.jp](mailto:i22110@oshima.kosen-ac.jp)

Teams：i22110（藤本宗太郎）