Raspberry Pi リモート開発環境の構築（公開資料） (Raspberry Pi Remote Development environment construction)			作成 2021/8/8 ニューラルソフト有限会社 市来 博記
初版	－	2021/8/8	
B	Visual Studio 2019 → Visual Studio 2022 に変更し、本書の記載で開発環境を構築できることを確認した。	2021/12/27	
C	ビルド イベント設定例を追加した。	2022/1/17	

概要

本書は、Windows 10 が稼働する PC に Raspberry Pi リモート開発環境を構築する方法について調査した結果を記したものです。

調査内容

Visual Studio 2022 のインストールから LAN に接続された Raspberry Pi 上の実行モジュールのデバッグ方法までを調査しました。

調査項目を示します。

1 Visual Studio 2022 のインストール
2 Windows PC と Raspberry Pi 間の SSH 接続
2.1 Windows 側の設定
2.2 Raspberry Pi 側の設定
3 リモート開発に必要なソフトウェアー
4 Visual Studio のクロス プラット フォーム接続の設定
5 プロジェクトの作成
6 プロジェクトのビルドとデバッグ
7 実行モジュールのマルチ スレッド化

情報源

Microsoft Site:Linux ワークロードのダウンロード、インストール、セットアップ
Microsoft Site:Azure の Linux VM に対する認証用に SSH キーを作成して管理する

前提条件

調査における前提条件を示します。

- 開発用ホスト マシンは、Windows10（64 ビット）がインストールされた PC（i7-3930K，32GB メモリ搭載の PC）を使用します。
- ターゲットの Raspberry Pi はモデル 3B(Raspberry Pi OS 32BIT)とモデル 4B(8GB Raspberry Pi OS 64BIT β)を使用します。
- Visual Studio 2022 は、無料版の Community を使用します。

- SSH 接続の認証方式は、公開鍵認証とします。
- 目標となる開発環境は、図 1 の通りです。

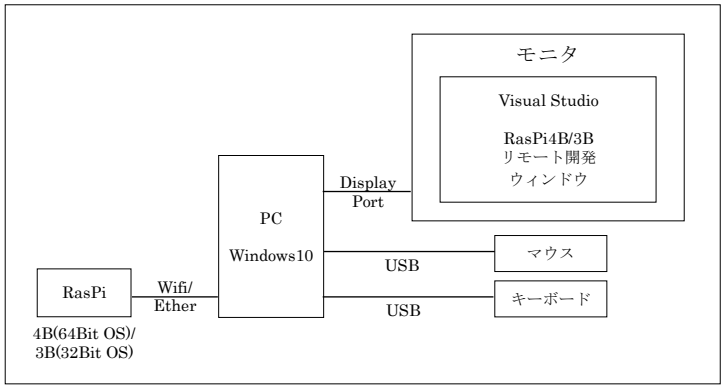

図 1 目標の開発環境

- 本書の記載内容は Microsoft Visusal Studio での C/C++言語プログラムのビルドとデバッグの経験がある方が対象です。

調査結果

〔Visual Studio 2022 のインストール〕

1 Microsoft のサイトの Visual Studio のページから Visual Studio 2022 Community のインストーラをダウンロードする。

2 インストーラを起動してワークロードを選択する。
「C++によるディスクトップ開発」と「C++による Linux 開発」のみチェック オンすると最小限セットアップになります。

3 右側のインストールの詳細で追加と除外のオプションを設定してインストールを進める。
＜設定例＞

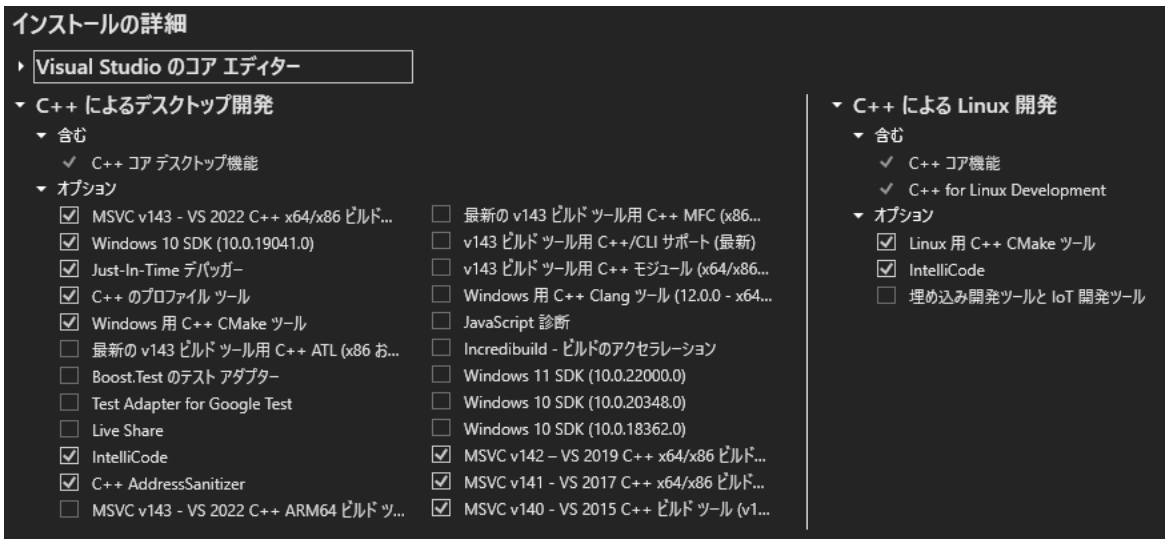

図 2 インストール オプション設定例

後は、インストールが完了するのを待つだけです。

〔Windows PC と Raspberry Pi 間の SSH 接続〕

〔Raspberry Pi 側の設定 1〕

sudo raspi-config コマンドで構成プログラムを起動して、Interface option/P2 SSH を選択して「YES（はい）」を選択します。

〔Windows 側の設定 1〕

1 次のコマンドで秘密鍵を生成する。
ssh-keygen -t rsa -m PEM -b 4096 -f 秘密鍵のファイル名 -N パスフレーズ
-f で指定した名称のファイルと".pub"が付加された名称のファイル（公開鍵）が生成されます。

2 コマンド プロンプトを開いて、次のコマンドで Raspberry Pi に接続する。
ssh Raspberry Pi のユーザ名@Raspberry Pi の IP アドレス（パスワード認証）
接続できない場合は、ネットワークを確認して下さい。

3 mkdir .ssh でディレクトリを生成する。

4 .ssh ディレクトリに移動後、次のコマンドを実行する。
vi authorized_keys

4.1 Windows のメモ帳などで ssh-keygen で生成した公開鍵ファイルを開いてテキストをコピーする。

4.2 コピーしたテキストを vi で開いている authorized_keys ファイルに貼り付ける。

4.3 vi の wq コマンドで書き込んだ後、vi を終了する。

5 exit コマンドで ssh を終了する。

6 次のコマンドで Raspberry Pi に公開鍵認証 SSH 接続する。
ssh -i 秘密鍵のファイル名 Raspberry Pi のユーザ名@Raspberry Pi の IP アドレス
パスフレーズの入力を求められますので、ssh-keygen で指定したパスフレーズを入力して下さい。

〔Raspberry Pi 側の設定 2〕

パスワード認証による SSH 接続を禁止する場合は、以下の設定を実施します。

1 /etc/ssh に移動する。

2 次のコマンドで sshd_config を開く
sudo vi sshd_config

2.1 #PasswordAuthentication yes → PasswordAuthentication no に変更する。

2.2 vi の wq コマンドで書き込んだ後、vi を終了する。

3 次のコマンドで sshd を再起動する。
sudo service ssh restart

〔Windows 側の設定 2〕

コマンド プロンプトで "ssh Raspberry Pi のユーザ名@Raspberry Pi の IP アドレス" を実行して、Permission denied が表示されれば、パスワード認証による SSH 接続が禁止されています。
SSH 接続時のパスフレーズ入力を省略する場合は、以下の設定を実施します。

1　Windows 10 の「コンピュータの管理」

1.1　左側リストで「サービス」を選択して、右側リストで OpenSSH Authentication Agent をダブルクリックする。

1.2　全般タブのスタート アップの種類を「自動」に設定する。

1.3　「サービスの状態」の「開始」ボタンを押下して、OpenSSH Authentication Agent を起動する。

2　コマンド プロンプトで次のコマンドを実行して、秘密鍵ファイルとパスフレーズの組み合わせを登録する。

ssh-add 秘密鍵のファイルのパス

パスフレーズの入力を求められますので、ssh-keygen で指定したパスフレーズを入力して下さい。

（秘密鍵ファイルとパスフレーズの組み合わせの登録を解除する場合：

ssh-add -d 鍵のファイルのパス）

〔リモート開発に必要なソフトウェアー〕

Visual Studio のリモート開発機能を利用するには、Raspberry Pi 側で以下のソフトウェアーが必要になります。

g++、gdb、make、ninja-build、rsync、zip

次のコマンドで必要なソフトウェアーをまとめてインストールします。

sudo apt install g++ gdb make ninja-build rsync zip

〔Visual Studio のクロス プラット フォーム接続の設定〕

Visual Studio のリモート開発機能を利用するには、Visual Studio に Raspberry Pi への接続の登録が必要になります。以下の手順で登録します。

1　Visual Studio 2022 を起動する。

2　メニューのツール/オプションを選択して、オプション設定画面を表示する。

3　左側リストで「クロス プラット フォーム/接続マネージャ」を選択して、「追加」ボタンを押下する。

4　リモート システムへの接続の設定画面で、次の項目を設定する。

4.1　ホスト名 ：Raspberry Pi の IP アドレス

4.2　ポート ：22（SSH サーバの設定で別のポート番号を指定した場合は、その番号を指定する。）

4.3　ユーザ名 ：Raspberry Pi のユーザ名

4.4　認証の種類 ：「秘密キー」

4.5　秘密キー ファイル ：ssh-keygen で生成した秘密鍵ファイルのパス

4.6　パス フレーズ ：ssh-keygen で秘密鍵ファイルを生成した時に指定したパス フレーズ

5　Raspberry Pi を起動して、ネットワーク状態を確認後、接続ボタンを押下する。

ssh-keygen コマンドで秘密鍵を生成する際に PEM 形式を指定していない場合、エラーになる可能性があります。（Visual Studio 2019 ではエラーになっていました。）

〔プロジェクトの作成〕

Raspberry Pi 用のプログラムを Windows でリモート開発するためのプロジェクトの作成は、コンソールアプリのプロジェクトの作成と変わらず、非常に簡単です。
プロジェクトの作成手順を以下に示します。

1 Visual Studio 2022 を起動する。
2 メニューのファイル/新規作成/プロジェクトを選択して、「新しいプロジェクトの作成」画面を表示する。
3 上部のコンボリストで、言語：C++、プラットフォーム：Linux、プロジェクトの種類：IoT を選択する。
4 表示されているプロジェクトの種類のリストから Raspberry Pi プロジェクトを選択して、「次へ」ボタンを押下する。
5 次の画面でプロジェクト名、格納場所、ソリューション名を指定して、「作成」ボタンを押下する。

上記の手順で作成したプロジェクトの雛形コードは、WiringPi ライブラリを使用した LED 点滅プログラムになっています。新しい GPIO へのアクセス方法として推奨されている libgpiod を使用するように対応する場合は、次のコマンドでライブラリとドキュメントをインストールします。

```
sudo apt install libgpiod2 libgpiod-dev libgpiod-doc
```

libgpiod 使用方法の例を以下に示します。（GPIO チップの全ラインの信号名称を表示した後、LED 点滅を 10 回行うプログラムです。）

```
#include <stdio.h>
#include <unistd.h>
#include <gpiod.h>

#define LED 17

int main(void)
{
	printf("テストだよ！¥n");

	//	チップのハンドルを取得
	struct gpiod_chip* chip = gpiod_chip_open("/dev/gpiochip0");
	if (chip != nullptr)
	{
		//	チップのライン数を取得
		int num = gpiod_chip_num_lines(chip);
		for (int i = 0; i < num; ++i)
		{
			//	ラインのハンドルを取得
			struct gpiod_line* line = gpiod_chip_get_line(chip, i);
			if (line != nullptr)
			{
				//	ラインの信号名を表示
				printf("line %2d: %s¥n", i, gpiod_line_name(line));
			}
		}

		//	LED 用のラインのハンドルを取得
		struct gpiod_line* ledLine = gpiod_chip_get_line(chip, LED);
		if (ledLine != nullptr)
		{
			//	LED 用のラインを出力に設定
			int sts = gpiod_line_request_output(ledLine, "testApp", 0);
			if (sts == 0)
			{
				for (int i = 0; i < 10; ++i)
				{
					gpiod_line_set_value(ledLine, 1);
					usleep(500 * 1000);
```

```
                        gpiod_line_set_value(ledLine, 0);
                        usleep(500 * 1000);
                    }
                }
            }

            gpiod_chip_close(chip);
        }
        return 0;
}
```

libgpiod を使用する場合、プロジェクトのリンク設定の -lgpiod の追加と、ビルド イベント設定のリモートのビルド後イベントの削除が必要です。

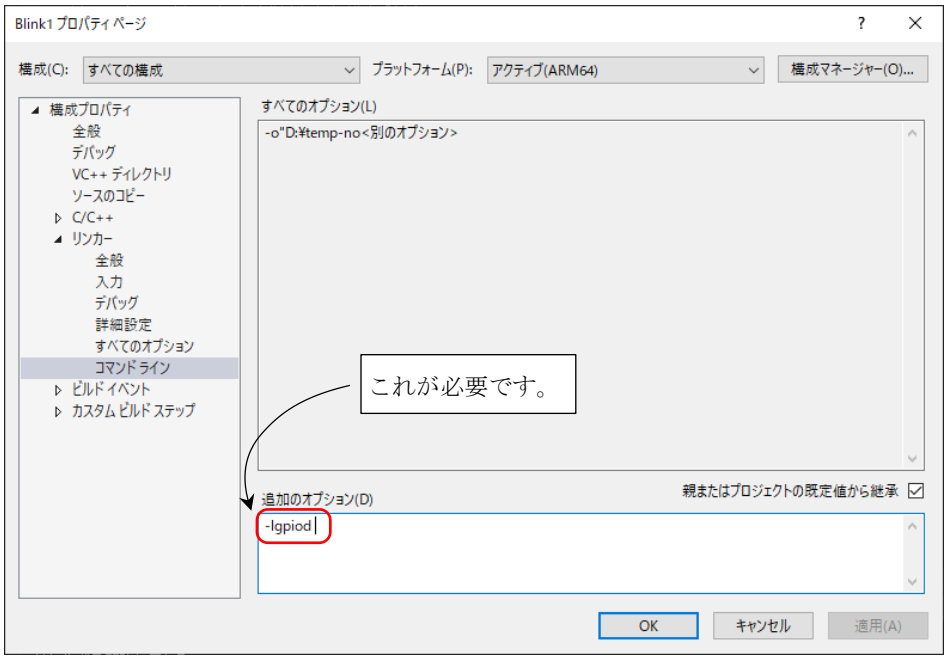

図 3 リンク オプション設定例

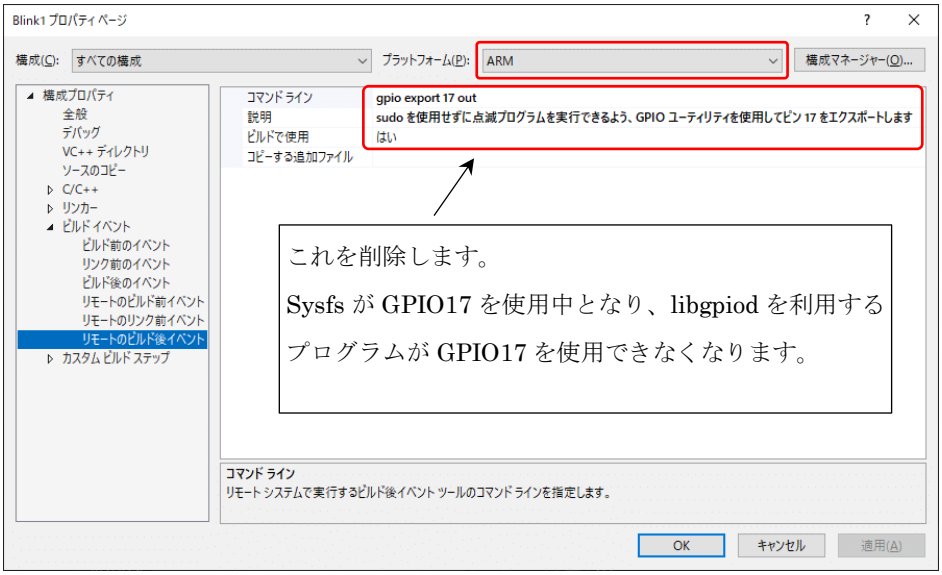

図 4 ビルド イベント設定例

〔プロジェクトのビルドとデバッグ〕

プロジェクトのビルドとデバッグの操作は、コンソール アプリ開発時と同じです。ユーザ プログラムのソース ファイルは自動的にターゲット(Raspberry Pi)に転送され、ターゲット上でビルドが実行されます。デバッグ開始・終了時も自動的にターゲット上の GDB と接続・切断され、コンソール アプリ開発時と同じ操作でデバッグできます。ターゲット プログラムのコンソール出力は、Visual Studio の Linux コンソール ウィンドウに出力されます。

〔実行モジュールのマルチ スレッド化〕

Raspberry Pi OS 上で動作するマルチ スレッド プログラムを開発する場合は、POSIX のスレッド モデル pthread を利用できます。（コンパイルとリンク オプションに -pthread の指定が必要です。）

所感

Raspberry Pi は一般的な OS で稼働させることができ、Window 環境もあるので、Visual Studio の linux 開発機能を利用すべきかどうか迷いましたが、十分に利用価値があることが実感できました。特にヘッドレスで稼働させるシステムの開発で有効だと思います。普通にインテリセンスが使えることや、キーボートとマウスを開発ホストと複数のターゲットで共有できるのもありがたいです。