

宇宙電子情報工学 2024 実習 (その 2)

福島 fukushima@isas.jaxa.jp

2024-01-22

概要

実習の後半では、前半（先々週）で設計した基板の実物を使用して、ソフトウェアの動作試験を行います。まず、KiCAD7で設計した基板の機能確認を行います。次に、その基板用にソフトウェアを準備し、それを動作させて搭載センサーからのデータ確認を行います。このソフトウェアは各自が生成 AI で生成していただき、基板上的 SBC（シングルボードコンピュータ）に投入してもらいます。今回の実習と前回の講義内容を合わせることで、簡単な組み込みシステム構築の一連の経験を得られることになります。今後、自分の研究活動で簡単な基板を準備する際、自分自身で準備することを選択肢の一つとして考えることができます。この実習内容はその作業のひな形、あるいは準備作業の出発点とすることができるでしょう。今回の実習作業では以下の作業を予定しています。

日付	作業 (成果物)	実習内容
2024/01/29	動作確認	1.基板チェック 2.プログラム準備と投入 3.機能チェック (UART/SPI/I2C)

1 基板チェック

1.1 基板単体チェック

1月15日の実習講義では、KiCADの操作について紹介しました。授業の構成上、時間的な制約から各操作を一つ一つ実際に行ってもらうことは難しかったため、興味を持たれた方が自分で作業するきっかけとなることを目指した実習でした。この日に構成した基板を基板発注しました。講義中に紹介したサイトを通じて、KiCADで生成したガーバーデータファイルをアップロードし、ネットで注文しました（5枚で3.5ドルでした）。

注文から4日後の1月19日金曜日に荷物が到着しました。詳細は1.2節で後述しますが、まず基板が壊れていないか確認する必要があります。荷物のパッケージにダメージがないか、基板に機械的な破損がないかをチェックします。海外からの輸送では荷物が破損することがよくありますが、基板は多少雑なハンドリングにも耐えることが多いです。

1. 基板に意図しない傷や割れがないかを確認する。
2. 目視で配線パターンの切れやビアの導通不良がないかを確認する。
3. 実体顕微鏡で配線パターン、ランド、ビアに意図しない穴がないかを確認する。
4. 配線パターンで接続されているはずの2つのランドにテスターのプロブを当てて導通をチェックする。

2年ほど前までは、配線パターンの品質に問題があって配線パターンが切れていたというようなこ

とはたまにありました。しかし、最近は品質が向上し、目視でわかるような不具合には遭遇していません。また、製造サービス会社によっては、オプションで機械的な導通チェックを行ってくれるところもあるので、必要であればそのように注文することも可能です。

1.2 部品実装と実装後チェック

基板にセンサーモジュールなどの部品をハンダ付けします。この基板では、ほとんど全ての部品をピンヘッダやピンソケットで基板に取り付ける方式を採用しています。そのため、部品にはピンヘッダをハンダ付けし、基板にはピンソケットをハンダ付けします。また、この基板には表面実装部品が4個使用されていますので、これらを直接基板にハンダ付けします。具体的には、I2C通信のラインに2個のチップ抵抗を、電源入力端子に2個のダイオードをハンダ付けします。表面実装部品のサイズは1608(1.6mm×0.8mm)であるため、実体顕微鏡を使用するハンダ付けが必要です。部品と基板上の位置については、KiCADを立ち上げてスキマティックと基板図を参照してください。

1. 部品を基板に取り付ける際には、上下を間違えないように注意する。
逆に取り付けると、電源が意図しない場所にかかり、部品が故障する可能性がある。
2. すべての部品を取り付けた後、外観チェックをする。
(ピンソケットでの接続なので、一つ一つ部品を取り付けての確認も可能)
3. 電源関連の端子について、電源の正負が逆になっていないかを確認する。
テスターを使用して、電源ラインが接続しているパッドが導通しているかを確認し、電源とGND間にKΩレベルの抵抗があること(ショートしていないこと)を確認。

1.3 電源投入チェック

その後、部品を搭載し、電源を投入します。最初に電源を投入する時が最も緊張する瞬間です。なぜなら、基板が壊れる可能性が最も高いのが、(経験上)最初の電源投入時だからです。電源投入後、各モジュールの電源端子について、所定の電圧がかかっているか(VCC-GND間が3.3V)を確認します。正しい値が出ていれば一安心です。

1. 電源投入後(USBコネクタから給電する場合)、回路上のVCC-GND間の電圧を確認します(5Vと3.3Vが存在します)。
2. 各モジュールのVin-GND間の電圧を確認します(3.3V)。
3. UARTのTX/RXが5V(場合によっては3.3V)、およびI2CのSCL/SDAが3Vであることを確認します。

2 プログラム準備と投入

2.1 Raspberry Pi Pico、Micropython、およびThonny

基板上のSBC(Raspberry Pi Pico、以下Picoと表記)にプログラムを投入します。Picoで動作するプログラム言語は複数存在しており、初心者にとって使いやすいのはArduino言語(C/C++とほぼ

同じ)か Micropython (Python のサブセット) です。今回は、より簡便な言語表記の Micropython を使用します。そして、Micropython でのプログラムを編集し、Pico に投入するためのツールとしてよく利用されている開発環境プログラムが *Thonny* です。今回の実習参加者は、上記のセットを各自の PC に準備してもらいます。具体的には、Pico 用に投入する Micropython のファームウェアと Thonny をネットワークから各自の PC にダウンロードしておいてもらいます。両者ともに無料です。

Pico 用ファームウェア	https://micropython.org/download/RPI_PICO/
Thonny	https://thonny.org/

2.2 Micropython のファームウェアアップデート

Pico に Micropython のファームウェアをアップロードします。これがないと Micropython のプログラムは動作しません。アップロード後、電源を投入すると Pico は Micropython の処理系として機能し、Micropython のプログラムによるピンの制御などが可能になります。PC と USB ケーブル (Pico は USB Micro : 最近までの Android で使用されていた端子) で接続します。

1. Pico のタクトスイッチを押しながら Pico を PC に接続します。
2. PC 上で USB ドライブとして Pico が認識されるので、そのフォルダに Pico 用のファームウェアをドラッグ&ドロップします。
3. Pico を PC から取り外し、タクトスイッチを押さずに PC に再接続すると、Pico は Micropython 処理系として動作します。

2.3 Thonny を使った Micropython プログラムの編集と実行

Pico 基板の準備が整ったら、Thonny でプログラムを編集し、Pico に送り込みます。Thonny はそのための機能が揃っています。大枠として、次の手順に従います。

1. Pico を PC に接続する。
2. Thonny を立ち上げ、【実行】 → 【インタプリタ設定】を進み、【インタプリタ】で Pico を選択する。
Pico はシリアルポート接続になっているので、それらしいものを選択する。
3. Thonny の編集画面にプログラムを書き込み、実行する。
4. Pico が動作する。

使い方やサンプルコードなどは、"micropython pico" で検索すればたくさんのサイトが見つかります。どのサイトにもチュートリアルが含まれており、ここで実施する内容ならばそれらを参照すれば独習可能でしょう。

```

from machine import Pin # 入出力モジュールを使用する
import time # タイマーモジュールを使用する

# 出力ピン設定
led = Pin("LED", Pin.OUT) # 本体LEDピンをledとして出力に設定

# 処理
while True: # 無限ループ
    led.value(1) # LEDを点灯
    time.sleep(0.5) # 0.5秒待つ
    led.value(0) # LEDを消灯
    time.sleep(0.5) # 0.5秒待つ

```

上記の指示に従い、Thonny の画面にコードを全てコピーして貼り付け、プログラム実行のボタンを押すことで、そのコードが実行されます。結果として、Pico 上で Micropython のプログラムが動作し、LED が点滅し始めることになります。

このプロセスは、Pico に Micropython を用いてプログラミングを行い、具体的な動作（この場合は LED の点滅）を確認するための基本的な手順を示しています。Thonny を介してコードを Pico に送り込むことで、プログラムの結果を直接的に視覚化し、動作確認を行うことができます。

2.4 プログラムの生成（GPT にコード生成させる）

生成 AI の代表格である ChatGPT を使用すれば、プログラム言語を知らなくても所望の機能を持たせたプログラムを入手できます。生成 AI の制約にもよりますが、GPT-4 という LLM を使用したものであれば、生成結果をそのままアップロードしても動作します。

ただし、生成 AI はハルシネーションという「間違い」を出力することがあり、出力結果をそのまま利用すると問題が起きます。この間違いは、悪意によるものではなく、「知らないとは回答したくないから、まあ適当に回答して先に進めちゃうかな」という感じのものです。AI の回答内容を理解できるのならば、「あやしいぞ」と気づけるようなものが多いです。また、出力結果が「大丈夫かどうか」を PC で実行して確認できるならば、プログラム言語を知らない人であっても生成結果であるプログラムを利用することは可能です。さらに言えば、初心者(学習中)の人が書いたプログラムよりも生成 AI の出力の方が(ずっと)信用できたりします(修正が必要な場合もありますが)。

AI に作業を要求するための入力を「プロンプト」といいます。ChatGPT でも Copilot でも、ブラウザからプロンプトを入力し、その回答がブラウザに表示されます。プロンプトに「これこれこういうプログラムをください」とすれば、結果としてプログラムが表示されます。そのプログラムをまるっとコピーして、Thonny の画面に貼り付け、プログラムを実行させれば Pico を動作させることができます。もしプログラムにエラーがあれば、Thonny から「プログラムにエラーがある」と表示されますし、あるいは、Pico が意図した挙動をしない(LED が光らない) ことになります。その場合は、プログラムがおかしいので、プロンプトの内容および出力結果であるプログラムを確認します。

プログラムをどのように引き出すか、その具体例を以下に示します。ChatGPT (GPT-4) への質問は、自分が最も知りたいことを直接入力するだけで十分です。そのままの質問をしたやり取りが次にあります。



あなた

micropythonのLチカのサンプルコードをください



ChatGPT

もちろんです。MicroPythonを使用してRaspberry Pi PicoでLEDをチカチカさせる（いわゆる"Lチカ"）ための簡単なサンプルコードを以下に示します。このコードは、PicoのGPIOピンに接続されたLEDを1秒間隔で点灯・消灯させます。

```
python Copy code

import machine
import utime

# GPIOピンの設定（例：GPIO 15にLEDを接続）
led = machine.Pin(15, machine.Pin.OUT)

while True:
    led.value(1) # LEDをオン（点灯）
    utime.sleep(1) # 1秒待機
    led.value(0) # LEDをオフ（消灯）
    utime.sleep(1) # 1秒待機
```

このコードは以下のように動作します：

1. `machine`モジュールと`utime`モジュールをインポートします。
2. 使用するGPIOピンを設定し、`machine.Pin`クラスを使用してLEDをそのピンに接続します。ここではGPIO 15を例としていますが、実際の接続に応じてピン番号を変更してください。
3. 無限ループ内で、`led.value(1)`と`led.value(0)`を使ってLEDを交互に点灯と消灯させます。
4. `utime.sleep(1)`は、LEDの状態変更の間に1秒間待機するために使用されます。

このコードをRaspberry Pi Picoにアップロードし、適切なGPIOピンにLEDを接続することで、LEDが点滅するのを見ることができます。もし異なる点滅パターンや速度を実現したい場合は、`sleep`関数の待機時間を調整してください。

プログラムだけが回答されるかと思いきや、プログラムコードの解説もしてくれます。それだけではなく、依頼していないにも関わらず、簡単な手順の解説もしてくれます。ChatGPTには、実に親切的な「人格」を感じてしまいます。ChatGPTを使い始めてから、福島は人（特に詳しい人、忙しい人、偉い人）に何かを問い合わせる気がなくなりました。ご自身で試されると分かりますが、どんな質問にも嫌な顔をせず（当然ですが）、必要な詳細さで回答してくれます。では、プログラムをどうやってPicoに入れるのか？プログラムをファイル（main.py という名称）にしてアップロード（USB

メモリと同じように、2.2で紹介) するのか？ ChatGPT に質問しました。



あなた

これをどうすればいいのでしょうか？ main.pyとしてセーブし、ドラッグ&ドロップするのでしょ



ChatGPT

はい、その通りです。Raspberry Pi PicoでこのLED点滅のコードを実行するためには、以下の手順に従います。

1. **コードの保存:** まず、提供したLED点滅コードをテキストエディタで開きます。Macでは「TextEdit」や「Visual Studio Code」などが使用できます。コードをコピーしてエディタに貼り付け、ファイル名を`main.py`として保存します。この名前は重要で、Picoは起動時に`main.py`を自動的に実行します。
2. **Raspberry Pi Picoにコードを転送:** Raspberry Pi PicoをUSBケーブルでMacに接続します。PicoがBOOTSELモードでない場合は、BOOTSELボタンを押しながら接続し、Picoがフラッシュドライブとしてマウントされるのを待ちます。
3. **ドラッグ&ドロップで転送:** Picoがフラッシュドライブとしてマウントされたら、先ほど保存した`main.py`ファイルをPicoのドライブにドラッグ&ドロップします。
4. **Picoの再起動:** ファイルの転送が完了したら、Picoを安全に取り外し、USBケーブルを一度抜いて再度挿し直します。この時、BOOTSELボタンは押さないでください。Picoは自動的に再起動し、`main.py`を実行し始めます。
5. **動作の確認:** Picoに正しくLEDが接続されていれば、Picoが再起動するとLEDが点滅し始めるはず

もし何か問題が発生した場合は、接続されているGPIOピンが正しいか、LEDが正しく接続されているか（正極と負極を間違えていないか）などを確認してください。また、Picoが正しくコン

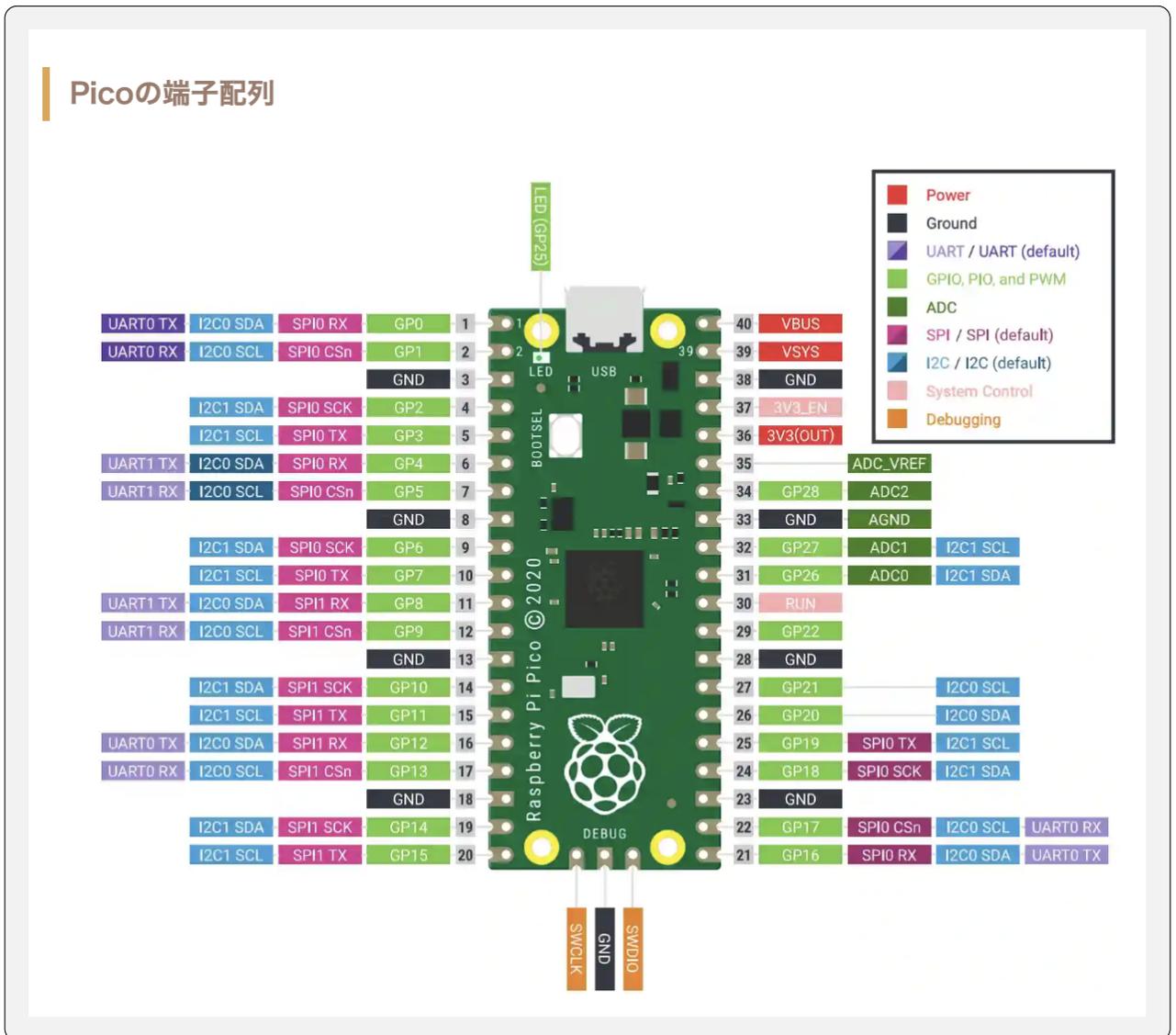
ピュータに認識されているかも確認することが重要です。これは通常、Macの「Finder」でPicoがフラッシュドライブとして表示されることで確認できます。

この手順に従ってもうまくいかない場合は、以下のことを試してみてください：

- Raspberry Pi Picoを別のUSBポートに接続してみる。
- 別のUSBケーブルを試す（一部のUSBケーブルは電力供給のみでデータ転送ができない場合があります）。
- Thonny IDEなどの開発環境を使用して、直接Picoにコードを書き込む。

これらのステップに従って、Raspberry Pi Picoで簡単なLED点滅プログラムを実行することができるはず

Raspberry Pi Pico のピン番号と機能 (SPI/UART/I2C) の一覧を次の表に示します。



テストとして、シリアルインターフェースのうち SPI で接続する装置のインターフェースプログラムを ChatGPT で生成します。まず、SPI0 を使用することにします (実習用基板では SPI1 を使いますので、基板では違うピンが使われています)。SPI0 で使用する 4 本のピン (SCLK, MISO, MOSI, CEn) の信号を Pico のどのピンに割り当てるかを指定します。

このとき、接続するモジュールを 9 軸センサとします。センサの型番と必要なデータ、データの更新レートが決まっていれば、それをプロンプトに含めます。これからプログラムで動かすデバイスの詳細情報については、入力すればするほど、こちらの要求に沿ったプログラムが生成されます。逆に、要求が曖昧な (抽象的である、あるいはどうしても解釈できる) 場合、生成されるプログラムも曖昧なものになります。プロンプトへの要求は、詳細であればあるほど、要求に合致したプログラムが生成されます。そして、生成 AI を上手に使うことはプロンプトをどう作るかという問題に帰着されます。いかにしてプロンプトを作るべきかという問題はすでに工学の一分野になっていて、それを「プロンプト・エンジニアリング」と呼びます。

MPU-9250 9 軸センサモジュール <https://strawberry-linux.com/catalog/items?code=12250>



あなた

rasberry pi picoでSPI0を使う。GP2, GP3, GP4, GP5のピンを使う。そして、動作速度は1Mとする。接続デバイスはMPU-9250で、角速度を1Hzで取得する。この場合のmicropythonのプログラムをください。



ChatGPT

Raspberry Pi PicoでSPI通信を用いてMPU-9250センサから角速度を取得するためのMicroPythonのプログラムを以下に示します。この例では、SPI0を使用し、GP2、GP3、GP4、GP5のピンを割り当て、SPIの動作速度を1MHzに設定します。

まず、MPU-9250センサとの通信にはSPIプロトコルを使います。MPU-9250は9軸センサで、加速度、ジャイロスコープ、磁気計のデータを提供します。このプログラムでは、ジャイロスコープ（角速度）のデータを1Hzの頻度で読み取ります。

ハードウェア接続:

- GP2 (SPI0 SCK)
- GP3 (SPI0 MISO)
- GP4 (SPI0 MOSI)
- GP5 (CS、Chip Select)

ChatGPTは、プロンプトで要求された項目をまとめ、プログラムを生成します。上記の例では、福島の要望に対してChatGPTが確認を行っています。飲食店で料理を注文を店員さんが復唱するのと同じです。このあとに、ChatGPTはプログラムを生成してそれを一気に出力します(次ページ)。

従来では、センサデバイスからデータをとりだすプログラムを作成するには、次の手順を踏んでいました。

1. 対象とするデバイスのデータシートをネットで検索し、それをダウンロードします。
2. デバイスへのアクセス方法についての注意を読み、それに沿ったプログラムを書きます。
3. その資料が英語だけの場合も少なくなく、また資料通りにプログラムを書き(たと思っても)ても希望通りに動かないことは普通でした。
4. 何度もトライアンドエラーを繰り返しながら、なんとかデバイスから必要なデータを取得するためのプログラムをまとめていきます。

有名なデバイスであれば、ネットで検索すれば誰かのブログがヒットしたり、GitHubあるサンプルコードが見つかったりします。しかし、使用している人が少ないセンサについては公開された情報が少なく、作業が行き詰まってしまうこともよくあります。今は、それらの苦労の大半をChatGPTが代行してくれます。作業全部とはいかないまでも、これまでトライアンドエラーをしていた部分が大幅にバイパスできます。しかもそれに必要な時間は1、2秒です。なんと素晴らしい時代でしょう！

プログラムは次のように出力されました。

プログラム:

```
python Copy code

from machine import Pin, SPI
import time

# SPIの初期化
spi = SPI(0, baudrate=1000000, sck=Pin(2), mosi=Pin(4), miso=Pin(3))

# CSピンの設定 (Chip Select)
cs = Pin(5, Pin.OUT)

# MPU-9250の初期化関数 (簡易版)
def init_mpu9250():
    cs.value(0) # CSをLOWに設定
    spi.write(b'\x6B\x00') # パワーマネージメント1レジスタをリセット
    cs.value(1) # CSをHIGHに設定
    time.sleep(0.1)

# 角速度の読み取り関数
def read_gyro():
    cs.value(0)
    # ジャイロスコープのデータレジスタ (0x43~0x48) を読み取る
    spi.write(b'\x43\x00')
    raw_data = spi.read(6)
    cs.value(1)
    # 生のデータを処理して角速度を計算 (適切なスケールリングが必要)
    gyro_x, gyro_y, gyro_z = int.from_bytes(raw_data[0:2], 'big'), int.
    return gyro_x, gyro_y, gyro_z

# MPU-9250の初期化
init_mpu9250()

# 1Hzの頻度で角速度を読み取る
while True:
    gyro_x, gyro_y, gyro_z = read_gyro()
    print("Gyro X:", gyro_x, "Gyro Y:", gyro_y, "Gyro Z:", gyro_z)
    time.sleep(1)
```

このプログラムは基本的な構造を提供しますが、MPU-9250の完全な初期化と正確なデータ解析には追加の設定が必要です。特に、生のジャイロスコープのデータはスケールリングとキャリブレーションを必要とします。

また、MPU-9250のデータシートや既存のライブラリを参考に、さらに詳細なセットアップやデータ解析の方法を調べることを勧めます。MicroPython環境でMPU-9250を使用する場合、既に存在するライブラリを利用することも可能です。

しかし、話はここで終わりません。ここでめでたしめでたしとはなりません。というのは、生成 AI にはハルシネーションが付き物だからです。生成結果が正しいとは限らないのです。あるいは、ChatGPT は最初から正しい情報を教えてくれるとは限りません。

Pico の SPI0 の位置からみて、ピン番号がおかしくないか???



あなた

Raspberry pi pico のGP2のPin番号はいくつですか？



ChatGPT

Raspberry Pi PicoのGP2のピン番号は1です。Raspberry Pi Picoでは、GP (General Purpose) ピンはGP0からGP22までの番号が付けられており、それぞれのGPピンには対応する物理的なピン番号が割り当てられています。GP2は物理的なピン番号1に対応します。



あなた

本当？



ChatGPT

申し訳ありません、私の前の回答は間違っていました。Raspberry Pi PicoのGP2は、物理的なピン番号4に対応しています。Raspberry Pi Picoのピン配置では、GP番号と物理的なピン番号が異なるため、注意が必要です。GP2は確かに物理的なピン番号4に位置しています。



あなた

再度プログラムをください。



ChatGPT

Raspberry Pi PicoでGP2、GP3、GP4、GP5のピンを使用してSPI通信を行い、MPU-9250センサから角速度を取得するためのMicroPythonのプログラムを再度記載します。ここで、GP2、GP3、GP4、GP5のピンはそれぞれ物理的なピン番号4、5、6、7に対応しています。

ハードウェア接続:

- GP2 (SPI0 SCK) → ピン番号4
- GP3 (SPI0 MISO) → ピン番号5
- GP4 (SPI0 MOSI) → ピン番号6
- GP5 (CS、Chip Select) → ピン番号7

おかしいと感じて ChatGPT を詰問すると、あっさりとは別の回答を出してきます。とても潔いのですが、それだからこそ新しい回答も「おかしいのではないか？」と心配になります。なので、結局どこからでも「生成されたプログラムを実行させて意図どおりに動作するか確認する」以外に方法はありません。したがって、「正しいかどうかは自分で資料を探して確認する」ことが求められます。

ここで、生成後の結果について ChatGPT とやり取りしながらプログラムを生成させた過程を紹介します。次ページから 4 例ほど示します。福島にとって ChatGPT は「作業担当者」であり、その人と口頭レベルでやり取りをしているだけです。この手のプログラムを自分で準備するとき、これまでは一人で作業してきましたが、これからは物知りな ChatGPT がアシストしてくれるわけです。

ChatGPT の生成結果のプログラムを実行しながら、その機能に対話的に変更していくことができます。何か問題がある場合は、その状況を ChatGPT に伝え、プログラムの修正を依頼します。修正されたコードを Thonny に貼り付け、Pico で実行して結果を確認します。この一連の作業を繰り返し、徐々に意図通りに動作するプログラムに近づけていきます。このプロセスでは、プログラムの変更はすべて ChatGPT による生成であり、ユーザー自身がコードを直接編集しませんでした（もちろんユーザーはいつでも好きに修正できます）。この作業を通じて、最終的にはゴールに到達させることが可能です。

また、次ページから示す例にある福島と ChatGPT のやり取りでは、プログラムだけでなくセンサの取り扱い方法についても議論(?) することができ、質問のタイミングに関わらず親切に回答を得ることが可能です。プログラムの実行結果を見て要求を変更することもでき、例えば、角速度データの追加、整数データを少数表示に変更する、サンプリング周期を変更するなど、さまざまな要望に応じて新たなコードを生成してくれます。

このような便利ツールの利用は、手軽に結果を得られる一方で、最終的にはプログラム言語の知識がある方が良い、センサの測定原理を知っている方が良いといった、基本的な知識の重要性に気づかせます。生成 AI の存在は、知らなくても済む知識をスキップしても生産性は向上する、という誤解を招くかもしれません。しかし実のところ、品質の高い結果を得るためには従来から必要な知識を学ぶ必要が出てきます。つまり、生成 AI があっても、学習する順番が前後するだけで、結局は学習が必要であるということに変わりはありません。



あなた

MPU-9250をSPIモードでデータ取得する場合の送信手順を具体的に教えてください。



ChatGPT

MPU-9250をSPIモードで使用してデータを取得する際の手順は、以下のようになります。SPI通信では、デバイスとの間でデータの読み書きを行うために特定のプロトコルに従います。MPU-9250の場合、このプロトコルはセンサのデータシートに基づいています。

基本的なSPI通信プロトコル:

1. **SPIインターフェースの初期化:** SPI通信を行うためのSCK（クロック）、MOSI（Master Out Slave In）、MISO（Master In Slave Out）、CS（Chip Select）ピンを設定します。
2. **チップセレクト（CS）のアクティブ化:** データ転送を開始する前に、CSピンをLOWにします。これにより、MPU-9250が通信に対してアクティブになります。
3. **レジスタアドレスの送信:** 読み取りたいレジスタのアドレスを送信します。SPIでの読み取り操作を示すために、アドレスの最上位ビット（MSB）を1に設定します。例えば、ジャイロスコープのデータが始まるアドレス0x43を読み取りたい場合、送信するアドレスは `0x43 | 0x80`（バイナリで `10000011`）になります。
4. **データの読み取りまたは書き込み:** レジスタアドレスを送信した後、次のバイトでデータを読み取ります。読み取りたいバイト数だけクロックを続け、各クロックで1バイトのデータを読み取ります。
5. **チップセレクト（CS）の非アクティブ化:** 通信が終了したら、CSピンをHIGHにして通信セッションを終了します。

MPU-9250でのデータ読み取り例:

MPU-9250からジャイロスコープのデータを読み取る例を示します。

```
python
```

Copy code

```
# CSをLOWにして通信開始
cs.value(0)

# ジャイロスコープデータ開始アドレス（読み取り操作を示すために0x80をOR演算）
spi.write(b'\x43\x80')

# 6バイト読み取り（ジャイロスコープデータはX、Y、Zの各軸で2バイトずつ）
gyro_data = spi.read(6)
```



あなた

加速度の場合を教えてください



ChatGPT

MPU-9250をSPIモードで使用して加速度データを読み取る場合の手順は以下の通りです。加速度データは、加速度センサーのX、Y、Z軸の各軸に対する値を含んでおり、通常は各軸について2バイト（16ビット）のデータが提供されます。

加速度データの読み取り手順:

1. **SPIインターフェースの初期化:** Raspberry Pi PicoのSPIインターフェースを初期化し、必要なピン（SCK、MOSI、MISO、CS）を設定します。
2. **チップセレクト（CS）をアクティブにする:** データ転送を開始する前に、CSピンをLOWにしてデバイスをアクティブ化します。
3. **加速度データのアドレスを送信:** 加速度データの開始アドレスを送信します。MPU-9250の加速度データはレジスタ0x3Bから始まります。SPI読み取り操作を指示するために、アドレスの最上位ビット（MSB）を1に設定します。つまり、送信するアドレスは `0x3B | 0x80`（バイナリで `10111011`）になります。
4. **データの読み取り:** 加速度データのレジスタアドレスを送信した後、6バイト分のデータ（X、Y、Z軸各2バイト）を読み取ります。
5. **チップセレクト（CS）を非アクティブにする:** 通信が終了したら、CSピンをHIGHにして通信セッションを終了します。

加速度データ読み取りの例:

```
python Copy code  
  
# CSをLOWにして通信開始  
cs.value(0)  
  
# 加速度データの開始アドレス（読み取り操作を示すために0x80をOR演算）  
spi.write(b'\x3B\x80')  
  
# 6バイト読み取り（加速度データはX、Y、Zの各軸で2バイトずつ）  
accel_data = spi.read(6)  
  
# CSをHIGHにして通信終了  
cs.value(1)  
  
# 加速度データの解析  
accel_x = (accel_data[0] << 8) | accel_data[1]  
accel_y = (accel_data[2] << 8) | accel_data[3]
```



あなた

`spi.write(b'\x3B\x80')`ではバイト送信していますか？



ChatGPT

``spi.write(b'\x3B\x80')``のコード行では、実際には2バイトのデータをSPIバス上で送信していますが、この方法はMPU-9250のデータ読み取りにおいて適切ではありません。正しくは、先にアドレスバイトを送信し、その後で必要なバイト数のデータを読み取る必要があります。

MPU-9250のSPI通信では、最初にレジスタアドレスを送信し、その後にデータを読み取ります。レジスタアドレスを示す際、最上位ビットを1に設定して読み取り操作を指示します。以下の手順で進めるのが正しいです：

1. **アドレスバイトの送信:** 最初に、読み取りたいレジスタのアドレスを送信します。この場合、加速度データは0x3Bから始まるので、0x3Bの最上位ビットを1にして、``0x3B | 0x80``をアドレスとして送信します。
2. **データの読み取り:** アドレスバイトを送信した後、SPIバスを介して必要なバイト数（加速度の場合は6バイト）のデータを読み取ります。

コードは以下のように修正されるべきです：

```
python Copy code

# CSをLOWにして通信開始
cs.value(0)

# 加速度データの開始アドレスを送信（読み取り操作を示すために0x80をOR演算）
spi.write(bytes([0x3B | 0x80]))

# 6バイト読み取り（加速度データはX、Y、Zの各軸で2バイトずつ）
accel_data = spi.read(6)

# CSをHIGHにして通信終了
cs.value(1)

# 加速度データの解析
accel_x = (accel_data[0] << 8) | accel_data[1]
accel_y = (accel_data[2] << 8) | accel_data[3]
accel_z = (accel_data[4] << 8) | accel_data[5]
```

このコードは、MPU-9250から加速度データを正しく読み取るための正確なSPI通信



あなた

これを加味して1HzでGyroデータを出力するプログラムを出し直してください。



ChatGPT

Raspberry Pi Picoを使用してMPU-9250からジャイロスコopデータをSPI通信で1Hzの頻度で読み取るためのMicroPythonプログラムを以下に示します。このプログラムでは、SPI通信の適切な手順を踏んで、ジャイロスコopデータを読み取ります。

プログラム:

```
python Copy code  
  
from machine import Pin, SPI  
import time  
  
# SPIの初期化  
# ここでは、PicoのGP2、GP3、GP4、GP5をSPIのSCK、MISO、MOSI、CSとして使用  
spi = SPI(0, baudrate=1000000, sck=Pin(2), mosi=Pin(4), miso=Pin(3))  
cs = Pin(5, Pin.OUT)  
  
# MPU-9250の初期化関数  
def init_mpu9250():  
    cs.value(0) # CSをLOWに設定  
    spi.write(b'\x6B\x00') # パワーマネージメント1レジスタをリセット  
    cs.value(1) # CSをHIGHに設定  
    time.sleep(0.1)  
  
# ジャイロスコopデータの読み取り関数  
def read_gyro():  
    cs.value(0)  
    # ジャイロスコopデータ開始アドレスを送信  
    spi.write(bytes([0x43 | 0x80]))  
    # 6バイト読み取り  
    gyro_data = spi.read(6)  
    cs.value(1)  
    # 生のデータを解析して角速度を計算  
    gyro_x = (gyro_data[0] << 8) | gyro_data[1]  
    gyro_y = (gyro_data[2] << 8) | gyro_data[3]  
    gyro_z = (gyro_data[4] << 8) | gyro_data[5]  
    return gyro_x, gyro_y, gyro_z  
  
# MPU-9250の初期化  
init_mpu9250()
```

上記の例を参考に、実習用の基板に装着されているデバイスからデータを取得するプログラムを生成し、それを Pico に送信して動作させます。Pico には複数のデバイスが装着されており、すべて同時に読み込むことが可能です。いきなり複数を取るとなんだかわからなくなるので、まずは一つ一つ動作させるプログラムを生成して最後に一つのプログラムに結合させます。複数のプログラムを一つのプログラムに統合するなどのソフトウェアの編集作業も ChatGPT で可能です。プロンプトにプログラムを与えて希望する指示をすればよく、結果としてプログラムが生成されます。

基板には次表のモジュールが連結されています。これを参考にそれぞれのデバイスを動作させるコードを生成します。ピン番号は KiCAD のスキマティックを表示し、Pico のどのピンに結合されているかを確認すればよいです。

	内容	型番	I/F	pin/line/SMD	備考
1	GPS/R	GT-723F	UART	バラ線	測位結果の NMEA テキスト
2	GYRO/ACC	MPU9520	SPI	2.54mm ピン	MEMS センサ出力(9 軸センサ)
3	16bit A/D	GY-ADS1115-4	I2C	2.54mm ピン	アナログ計測
4	USB シリアル	AE-FT231X	UART	2.54mm ピン	PC との通信
5	LED		DO	表面実装	状態表示
6	SBC	RaspberryPiPico		2.54 ピン	

3 機能チェック

プログラムが投入された Pico を動作させ、所望のデータが出力されるかを確認します。角速度・加速度を出力する MPU-9250 が一番わかりやすいです。しかし、その他のセンサも「何らかのデータ」が出力されていることを確認できればよいです。センサ情報がどこまで正しいかを吟味するには、それなりの時間がかかります。センサのキャリブレーションと呼ばれる工程を経るからです。この実習では、そこまでは立ち入りません。

ここで実施したもの以外のセンサでも、大抵のものは生成 AI (ChatGPT Windows-copilot GIT-copilot など) でプログラムを生成してくれます。利用者が多いセンサほど、詳細で洗練されたプログラムが生成される傾向にあります。一方で、使用者が少ないセンサの場合には、実用に耐えないようなプログラムを出力することもあります。

生成 AI が出力するプログラムの品質にはばらつきがあり、現時点では利用者の鑑識眼が求められます。しかし、このばらつきは時間の問題だと思われます。現在問題となっていることは、当然 AI 開発者側も認識しているでしょうから、競争が進むにつれてそのうち解消されるでしょう。

なお、この講習でのプログラム生成ついでの見解は 2024-01-22 時点のものであり、3 か月後には大きく変わる可能性があります。しかし、サンプルプログラム程度のものを生成 AI に作らせるという手順は、今後より標準的(あたりまえ)なことになると想像しています。プログラム生成がどこまで進化するかはわかりません。サンプルを超えてどの程度の実利用が可能になるのか、自動運転技術みたいな扱いと同じものでしょう。生成 AI の利用については多くの意見がありますが、結局のところ、これらは道具ですので、上手く使って自分の目指すことを実現させることが大切です。その態度自体には、今後も大きく変える必要はないでしょう。