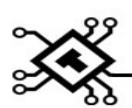


バーサライタを作ろう (ESP32バージョン)

本書はバーサライタを作るための手順を順に説明したものである。

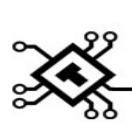
文責 滝田謙介 E-mail:takita@nit.ac.jp





目次

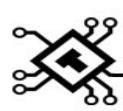
- 1. 概要**
2. 組み立て
3. Lチカに挑戦
4. タイマをつかう
5. センサを読み込む
6. センサとタイマを使った角度計算
7. LEDのスタティック点灯・ダイナミック点灯
8. 文字データの表示
9. 好きな文字を表示させてみよう



バーサライタ・概要

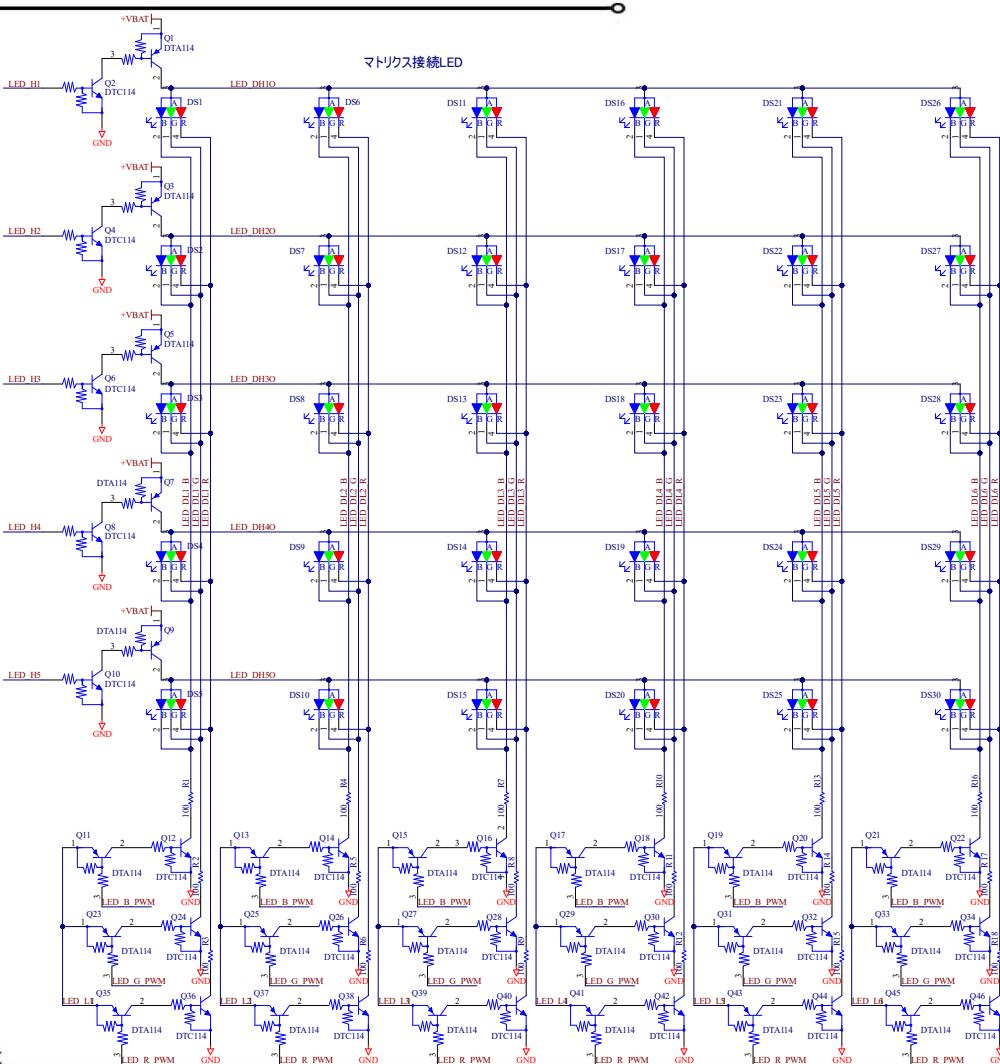
- ・バーサライタは、目の残像を利用して、空間に文字や絵を描くためのものである。
- ・ロボットの制御に必要な時間の管理を学ぶための教材でもある。
- ・作成するバーサライタは、LEDを30個を搭載して高さ(または幅)30ドットの絵・文字を描くことが出来る。
- ・姿勢センサ(ジャイロセンサ)と32ビットのIoTマイコンと言われるESP32を搭載している。
- ・ジャイロセンサから出力される角速度を取得し、時間係数をかけて足し算をすることで、角度のデータを作る。
- ・32ビットマイコンでは、角度のデータから表示するデータを選択し、LEDを点灯させる。
- ・LEDの駆動方法には、スタティック点灯(静的点灯)ではなく、ダイナミック点灯(動的点灯)を採用し、省電力化を図っている。
- ・USBシリアル変換回路を搭載しているのでUSBがあれば、プログラムが可能。

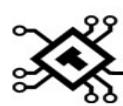




回路図(秋)(単色版)

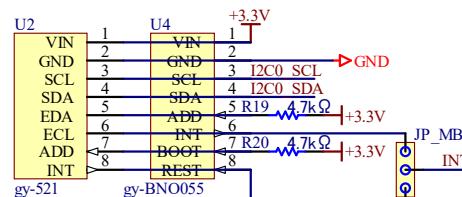
- マトリクス接続されたLEDを駆動するためのトランジスタアレイをCPUで制御する回路構成になっている。
- ソース側(電流供給側)を5組、ソース側(電流吸入側)を6組でアレイを構成している。
- ソース側は、RGBの三系統で構成している。
- NPN(プラスでON) + PNP(マイナスでON)の二つのトランジスタでスイッチ回路を構成している。
- LEDはダイナミック点灯とし、回路の簡素化・消費電力の低減を図っている。





回路図(秋)(単色版)

- I2C接続の各種姿勢センサ(ジャイロセンサ)を接続して、角速度・加速度などを取得する。(製作例:ジャイロセンサから取得した角速度データから角度を計算する。)
 - 実習用に、弾丸型LEDを1個、半固定抵抗などを備えている。
 - センサはジャンパ接続を変更することで対応している。(MPU625x(gy-521), BNO055(gy-BNO055またはAE-BNO055-BO), AE-L3GD20に対応)。
 - CPUとしてESP32 DEVKITCを搭載。

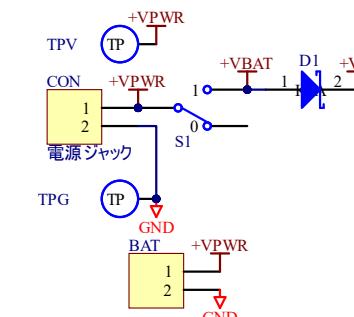
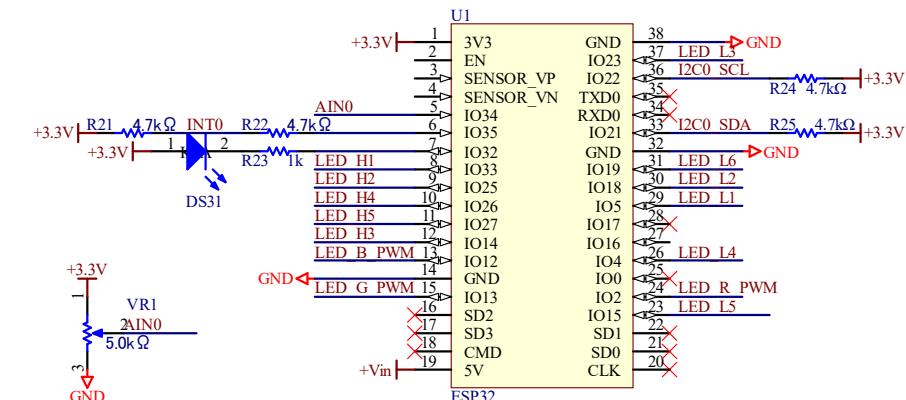
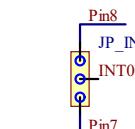
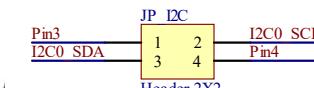
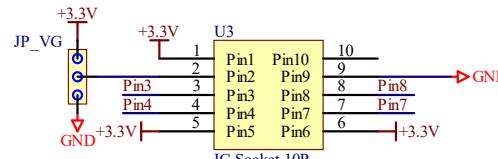
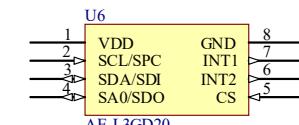
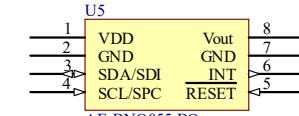


gy-521 接続時

gy-BNO055 接続時

- U3 の 上側を使用
- JP_VG GND側をショート
- JP_INT Pin8側をショート
- JP_I2C Pin3-SDA,Pin4-SCL

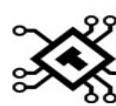
- ・ U3 の 下側を使用
- ・ JP_VG 3.3V側をショート
- ・ JP_INT Pin7側をショート
- ・ JP_I2C Pin3-SCL Pin4-SDA





部品表

部品番号	製品型番	説明	量	値	秋月通販コード
BAT	BH-321-1A-150 2個	電池接続用パッド	1		P-17271
CON	MJ-179PH	DCジャック	1		C-06568
D1	SB240LES			12A Vf:0.41V	I-16419
DS1, DS2, DS3, DS4, DS5, DS6, DS7, DS8, DS9, DS10, DS11, DS12, DS13, DS14, DS15, DS16, DS17, DS18, DS19, DS20, DS21, DS22, DS23, DS24, DS25, DS26, DS27, DS28, DS29, DS30	OSTAMA5B31A	RGBフルカラーLED 大きさ 5mm アノードコモン	30フルカラ		I-12168
DS31	指定なし	Φ3mmのLED (色は好みで)	1		例えばI-09851
JP_I2C	2x2 ピンヘッダ	ジャンパーピン	1		C-00080 または C-00167
JP_INT, JP_MB, JP_VG	1x3 ピンヘッダ	ジャンパーピン	3		C-00167
Q1, Q3, Q5, Q7, Q9, Q11, Q13, Q15, Q17, Q19, Q21, Q23, Q25, Q27, Q29, Q31, Q33, Q35, Q37, Q39, Q41, Q43, Q45	DTA114EL	抵抗入り PNPトランジスタ	23		I-12464
Q2, Q4, Q6, Q8, Q10, Q12, Q14, Q16, Q18, Q20, Q22, Q24, Q26, Q28, Q30, Q32, Q34, Q36, Q38, Q40, Q42, Q44, Q46	DTC114EL	抵抗入り NPNトランジスタ	23		I-12467
R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R22	1/2W カーボン抵抗 または 1/4W カーボン抵抗	抵抗	18 100Ω		R-07803 または R-25101
R19, R20, R21, R24, R25	1/4W カーボン抵抗	抵抗	64.7kΩ		R-25472
R23	1/4W カーボン抵抗	抵抗	11k		R-25102
S1	SS22F06G5-G	スイッチ	1		P-03252
U1			1		M-15673
U2	GY-521 または GY-BNO055	姿勢センサ	1		
U3	AE-BNO055-BO または AE-L3GD20	姿勢センサ	1		K-16996 または K-15096
VR1	3362P-1-503LF	半固定抵抗	1		P-14905 など

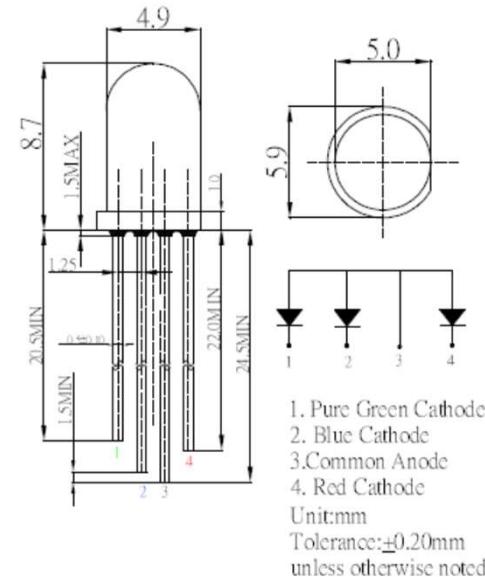


LEDの電流制限抵抗(R1～R18)

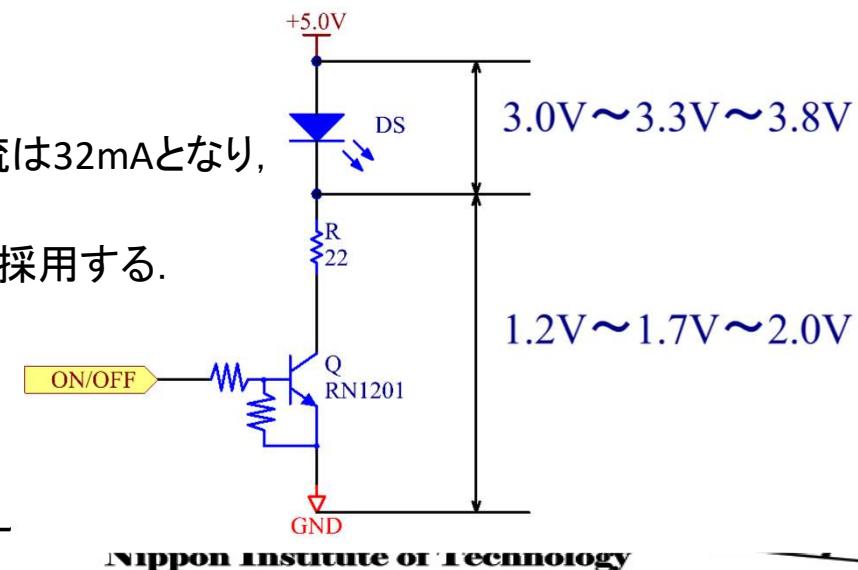
- LEDは様々な特性を持っているが、回路を設計する上で重要なパラメータが、順方向電圧と順方向電流である。
- 特に順方向電流は最大値を知ることが壊さないで使用するために重要である。
- 今回使用するフルカラーLED(OSTAMA5B31A)の最大順方向電流および順方向電圧は、30mA(赤)・50mA(青・緑)と1.8～2.6V(赤), 2.8～3.6V(青・緑)である。
- 電源電圧を5Vとすると、抵抗とトランジスタに印加される電圧が2.4V～3.2V(赤) / 1.4V～2.2V(青・緑)となる。トランジスタのV_{ce}(コレクタ・エミッタ間電圧)は、0.1Vぐらいなので、ここでは無視する。

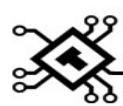
赤色LEDについて考える。

- 2.4Vの時に、30mAとすると、 $2.4V/30mA \approx 80\Omega$, $2.4V \times 30mA = 72mW$
- 3.2Vの時に、30mAとすると、 $3.2V/30mA \approx 107\Omega$, $3.2V \times 30mA = 96mW$
- 仮に100Ωを選ぶとすると、LEDの順方向電圧が低い場合、LEDに流れる電流は32mAとなり、定格の最大絶対30mAを超えててしまう。
- しかし、トランジスタなどでの電圧降下もあるので、抵抗値としては、100Ωを採用する。
- 青・緑の抵抗についても同様に計算して決める。
- 最悪値を考えるならば、150Ωの抵抗にしておく。



データシートより





動作確認

- Arduino IDEをインストールして、以下のサイトを参考にESP32をボードマネージに登録する。

<https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html>

※ ファイル自体は、下記のサイトを参照。

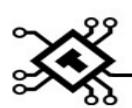
<https://github.com/espressif/arduino-esp32>

- パソコンなどのUSB端子にMicroUSB Type Bのコネクタを持ったケーブルのTypeA側を差し、MicroUSBをESP32 DevkitCのUSB端子に接続する。
- 必要ならば、DevkitC搭載のICのドライバをSilicon Labs社のWEBからダウンロードしてインストールする。（<https://jp.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>）
- テスト用プログラムをArduino IDEからESP32に書き込みLEDの駆動が正常に行なわれることを確認する。

※ テスト用プログラムは、下記のURLを参照する。

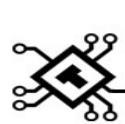
https://github.com/blues27/ESP32_POV_LightStickV4/tree/main/VERSAWRITER_L3GD20H_TEST





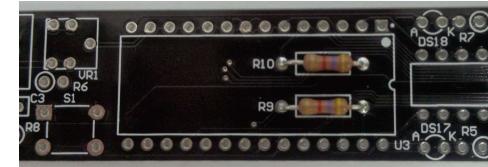
目次

1. 概要
2. 組み立て
3. Lチカに挑戦
4. タイマをつかう
5. センサを読み込む
6. センサとタイマを使った角度計算
7. LEDのスタティック点灯・ダイナミック点灯
8. 文字データの表示
9. 好きな文字を表示させてみよう

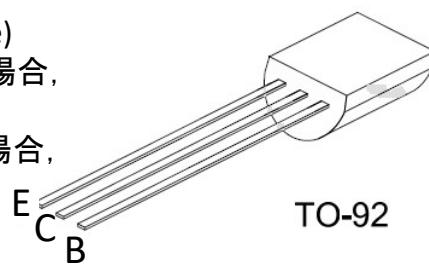
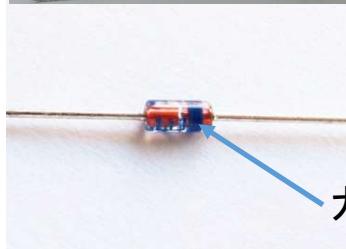


組み立て(1)

- ・基板に書かれた記号と部品表の番号を対応させて部品を載せていく。
- ・背の低い部品から基板にはんだ付けするのが定石。
- ・抵抗・積層セラミックコンデンサには極性がないのでどちら向きでもよい。
- ・ICの下に配置するコンデンサ C3, 抵抗 R9, R10は, ICソケットよりも先に基板につけるのを忘れないように。
- ・トランジスタ極性があるので取り付ける向きに注意する。(部品という番号が見えるようにいて, 左からECB)
- ・ダイオードは極性がある。基板上の○がついている穴がカソード端子がささる穴である。
- ・立ててはんだ付けするため, 黒い線のある方がカソードなので, 反対側(黒い線がない側)の足を曲げる。

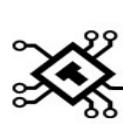


(E: Emitter, C:Collector, B : Base)
デジタルトランジスタ(NPN)の場合,
EはGND, Bは入力, Cは出力
デジタルトランジスタ(PNP)の場合,
EはVCC, Bは入力, Cは出力



Nippon Institute of Technology

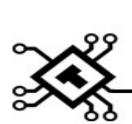




組み立て(2)

- ICソケット(U2, U3, U4)を取り付ける。向きがあるので、注意すること。基板上の印刷にあわせて搭載する。
- センサ(U2)用のソケットは、丸ピンのものを使用する。
- マイコン用のソケットは、丸ピンでも平ピンでもかまわない。
- 3mmのLED3個(DS17～DS19)を取り付ける。色は好みで選ぶ。極性があるので向きに気をつけること。足の長い方がアノード(A)、短い方がカソード(K)である。
- 半固定抵抗(VR1)を取り付ける。半固定抵抗にはいくつか種類があるため、使わない穴がある。

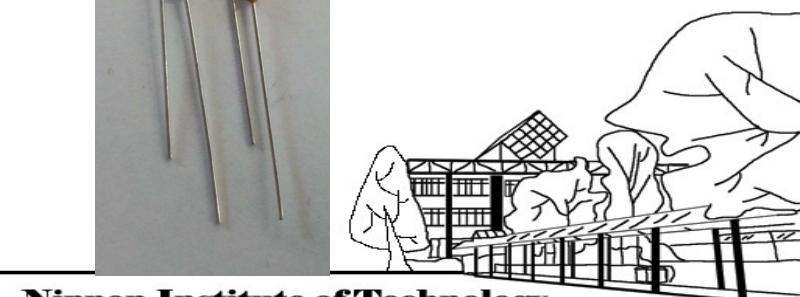


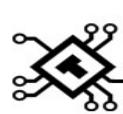


組み立て(3)

- LED(DS1～DS16)を取り付ける.
- LEDの向きをよく確認すること.
- LEDによっては基板上のシルクと部品の切り欠きが一致しないこともある.
- 16個のLEDが傾かないようにつけること.

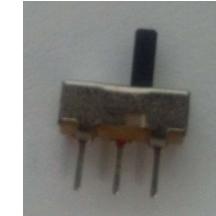
- 抵抗の足を曲げて、取り付ける.
- LEDの電流制限抵抗を取り付ける.
- R5, R7, R8は、3mmのLED(DS17～DS19)の電流を制限するための抵抗である.
- R6は、入力・出力を間違ったときに、IOピンを保護するための抵抗である.





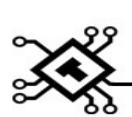
組み立て(4)

- タクトスイッチを取り付ける。
- スライドスイッチを取り付ける。
- スイッチが傾かないように、一つのpinを軽くハンダ付けし、指で押さえながら、はんだごてをあてて、ハンダをとかして、傾きなどを調整し、調整が終わったら、はんだごてを離すようにするとやりやすい。



- 3端子レギュレータ(U4)を取り付ける。
向きがあるので基板上のマークと一致するように挿入する。
- 電解コンデンサを取り付ける。横に寝かせるとよい。
- USBシリアルコンバータ用ピンヘッダを取り付ける。
- ピンヘッダは基板の裏・表のどちらの面でもよい。
- DCジャックを取り付ける。
- S5のスイッチを取り付ける。

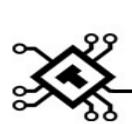




組み立て(5)

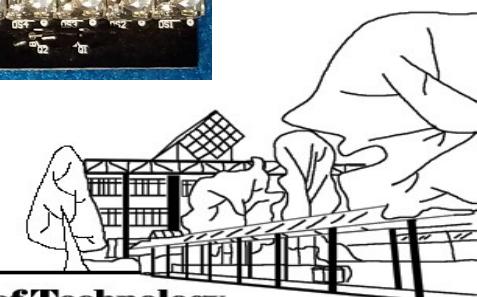
- 最後に、バーサライタ基板に載せるスイッチ、USBシリアル基板用ヘッダーピン、ジャイロセンサ基板などをはんだ付けする。
- USB基板は、ピンソケットを基板の上または下に取り付ける。バーサライト基板にも合うようにピンヘッダをはんだ付けする。
- センサ基板もピンソケットを基板の上または下に取り付ける。

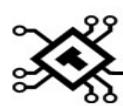




組み立て

- ・電池ケースは、基板の裏に取り付ける。(ネジ留めする場合は要スペーサ、または両面テープ)
- ・もしあれば、基板の裏側にスポンジなどのテープを貼り、ハンダがついた端子が手に刺さらないように工夫する。
- ・また、ジャイロセンサが、むき出しのため、CPU、ジャイロのあたりに手が当ると場合によってはショートする可能性があるので、周辺に布などをまき、手が直接当ることがないようにするとともにグリップを構成する。

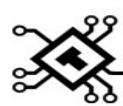




動作確認

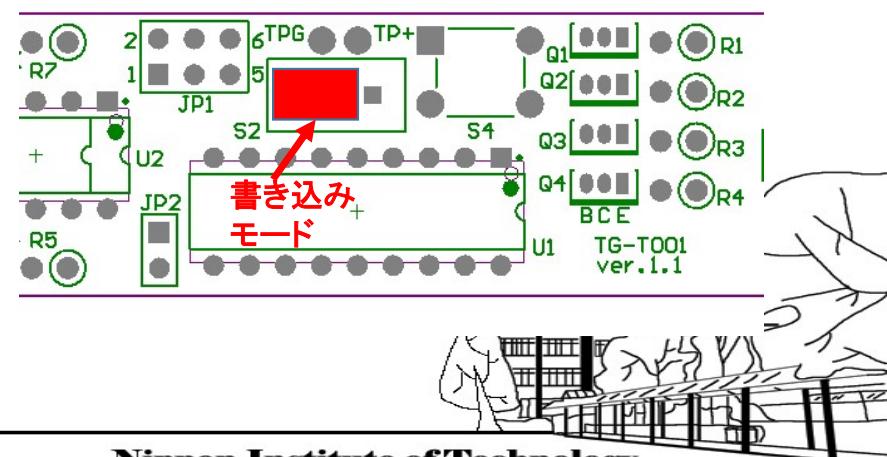
- ・パソコンなどのUSB端子にMicroUSB Type Bのコネクタを持ったケーブルのTypeA側を差し, MicroUSBをUSBシリアル基板のUSB端子に接続する.
- ・USB基板搭載のICのドライバをFTDI社のWEB(<http://www.ftdichip.com/Drivers/VCP.htm>)からダウンロードしてインストールする.
- ・5VのACアダプタの端子をバーサライタ基板裏側のDCジャックに挿す. 握して発熱している部品がないことを確認し, TP+—TPG間の電圧が3.3Vになっていることをテスターなどで確認する.
- ・テスト用ファームウェアをダウンロードし, LEDの駆動が正常に行なわれることを確認する.
- ・焼き込みに使用するソフトウェアは下記のURLからダウンロードする.
- ・http://www3.big.or.jp/~schaft/hardware/zip/LPCISP_010.zip
- ・使い方は下記のURLを参照.
- ・<http://www3.big.or.jp/~schaft/hardware/tips/LPC1114/page002.html>
- ・テスト用ファームウェアは, 下記のURLからダウンロードする.
- ・http://www.takitagiken.com/~takita/VersaWriter/K18_all_NIT_LPC1114_I2C.bin フルカラー
- ・http://www.takitagiken.com/~takita/VersaWriter/K18_all_NIT_LPC1114_SPI.bin 単色版

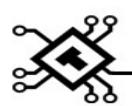




LPCISP

- バーサライタ基板をPCと接続し, S2を書き込みモードの位置(S2の文字のある側)にして, リセットボタンS1を押す. これでプログラムモードに切り替わる.
- 前述のURLよりLPCISPをダウンロードに起動する.
- binファイルには, ダウンロードしたbinファイルを指定する.
- COM番号は, Windowsのデバイスマネージャ等で確認しておく.
- 「書き込む」ボタンを押すと書き込みが行なわれ, 終了したらUSBケーブルを抜き, 電源を挿す.
- S2を実行モードの位置(S2の文字の反対側)にして, リセットボタンS1を押して, DS1～DS16までのLEDが一瞬点灯→消灯し, DS17, DS18のLEDが点滅→点灯したら, 正常に動作している. 振ると設定した「日本工業大学」が残像として表示される.





目次

- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- 8. LEDのスタティック点灯・ダイナミック点灯
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう

mbed登録(1)

- mbedのサイトの下部にある  ボタンをおして、



Logos of various mbed partners and supporters, including: i248, accenture, ADVANTECH, Alcatel-Lucent, ANALOG DEVICES, ATHOS, Atmel, Baidu, CYPRESS, and DELTA.

Logos of various mbed partners and supporters, including: element14, ERICSSON, ESPOTEL, expresslogic, FORGEROCK, GreenPeak, IBM, maxim integrated, MegaChips, and MULTITECH.

Logos of various mbed partners and supporters, including: myotest, NORDIC, nuvoTon, NXP, ON, POT, PollenTech, RDA, RENESAS, and salesforce.

Logos of various mbed partners and supporters, including: Schneider Electric, SEMTECH, SILICON LABS, SilverSpring, SK telecom, SmeshLink, SPANSION, SpinDance, ST, and SWITCHSCIENCE.

Logos of various mbed partners and supporters, including: ThunderSoft, TEXAS INSTRUMENTS, u-blox, WD, WIREPAS, WIZnet, wot.io, ZEBRA, and ZUMTOBEL.

Sign up for news from the mbed team: [Sign Up](#) [More information](#)

Language: [English](#) | [简体中文](#)

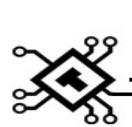
[About mbed](#) [Technologies](#) [Partners](#) [Development](#)

[What is mbed?](#) [Connectivity](#) [Our Partners](#) [Hardware](#)
[ARM mbed Enabled](#) [Security](#) [Become a Partner](#) [Software](#)
[Jobs](#) [Demonstrations](#) [Partner Portal](#) [Cloud](#)
[Events](#) [Applications](#) [Getting Started](#)
[Blog](#) [Community and Help](#)
[Documentation](#)

[LinkedIn](#) [YouTube](#) [Events](#) [Forum](#) [Blog](#)

ARM, Cortex, Keil, mbed, Thumb, TrustZone, ULINK, µVision, Versatile are trademarks or registered trademarks of ARM Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. [Further Detail](#)

© ARM Ltd. Copyright 2016 – ARM mbed IoT Device Platform [Home](#) | [Terms](#) | [Privacy](#) | [Cookies](#) | [Sitemap](#)



mbed登録(2)

- mbedのサイトの下部にある **mbed Classic Developer site** ボタンをおして、右図の開発者用ページに移動する。
- **Compiler** をおすと、

ARMmbed

Hardware ▾ Documentation ▾ Code Questions Forum Log In/Signup Compiler Search

Search developer.mbed.org...

ARM mbed Developer Site

mbed simplifies and speeds up the creation and deployment of devices based on ARM microcontrollers.

The project is being developed by ARM, its Partners and the contributions of the global ARM mbed Developer Community.

Explore mbed »



220,280 compilations in the last 7 days

Blog

- A new Bluetooth library: SimpleBLE
- Post-mortem debugging with ARM mbed
- Summer mbed Events in Taiwan and China
- Now available: Secure WebSockets and MQTT over TLS libraries
- Building your own LoRa network

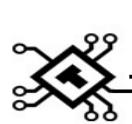
Questions

- string convert to int
- No response from SD card on FRDM-K64F
- mktime bug for year 2038?
- Equivalent to Arduino millis()
- Is LBT implemented in libmdot?

Activity

» Your dashboard

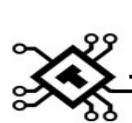
- New program: MCP23009tst - test program for the MCP23009 ...
- wimbeaumont Project - 22 minutes ago
- Library updated: MCP23009 - added more functions, tested ...
- wimbeaumont Project - 26 minutes ago
- Program updated: Water Play - Implemented SalinityController and TemperatureController
- Mike Triu - about 2 hours ago
- New program: M0_Test_7segment - M0
- Fabio Moretti - about 2 hours ago
- Program updated: Saltware_1 - Commit
- Jean de Wach - about 5 hours ago



mbed登録(3)

- mbedのサイトの下部にある **mbed Classic Developer site** ボタンをおすて, 右図の開発者用ページに移動する.
- **Compiler** をおすと, ログイン/ユーザー登録画面に切り替わる.
- **Signup** を押すとユーザー登録画面に切り替わる.

The screenshot shows the ARM mbed developer site's login and signup interface. The top navigation bar includes links for Hardware, Documentation, Code, Questions, Forum, Log In/Signup, and Compiler. A search bar is also present. The main area features two forms: a 'Login' form on the left and a 'Signup' form on the right. The 'Login' form fields are 'Username' (takita@nit.ac.jp) and 'Password' (redacted). Below these are links for 'I've forgotten my username' and 'I've forgotten my password'. A 'Remember me' checkbox is available. The 'Login' button is orange. The 'Signup' form has a large 'mbed' logo and an orange 'Signup' button. At the bottom of the page, there is a footer with links for mbed, blog, we're hiring!, support, service status, privacy policy, terms and conditions, and language selection (en ja es de).



mbed登録(4)

- mbedのサイトの下部にある **mbed Classic Developer site** ボタンをおすて, 右図の開発者用ページに移動する.
- **Compiler** をおすと, ログイン/ユーザー登録画面に切り替わる.
- **Signup** を押すとユーザー登録画面に切り替わる.
- 以前に登録したことがあるかと聞かれるので,

ARM mbed

Hardware Documentation Code Questions Forum Log In/Signup

Compiler

Create a developer.mbed.org account

Search developer.mbed.org... Search

Let's get started!

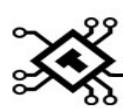
Just one question first, though.

Have you ever signed up on developer.mbed.org before?

Yes, I have created an account before

No, I haven't created an account before

© mbed | [blog](#) | [we're hiring!](#) | [support](#) | [service status](#) | [privacy policy](#) | [terms and conditions](#) | Language: [en](#) [ja](#) [es](#) [de](#)



mbed登録(5)

- mbedのサイトの下部にある **mbed Classic Developer site** ボタンを押し、右図の開発者用ページに移動する。
- **Compiler** をおすと、ログイン/ユーザー登録画面に切り替わる。
- **Signup** を押すとユーザー登録画面に切り替わる。
- 以前に登録したことがあるかと聞かれるので、

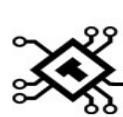
No, I haven't created an account before

を選ぶと、Signup画面になる。ここで、必要な項目を入力してアカウントを作る。

Complete the form below to sign up to mbed!

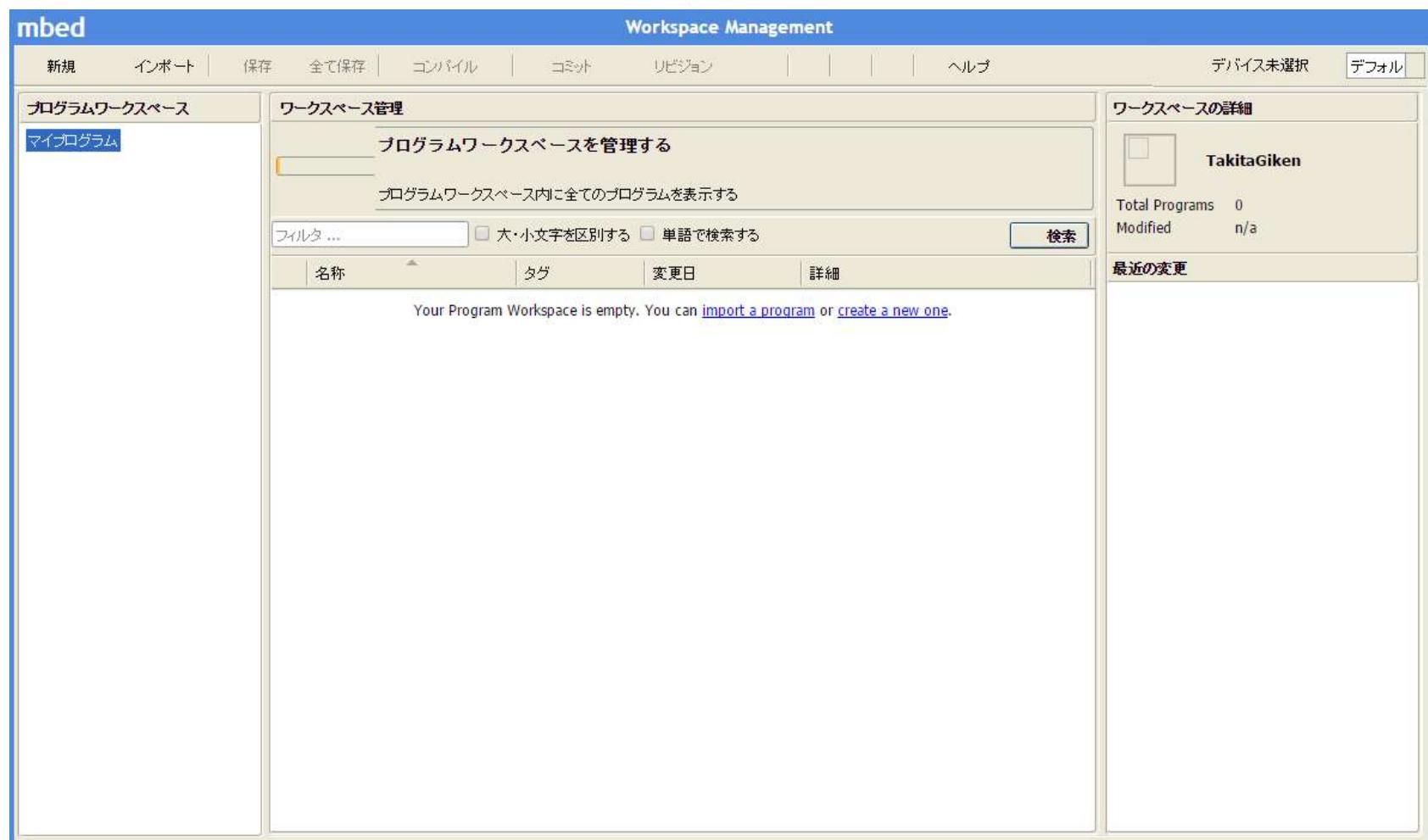
Signup	Summary
あなたのメールアドレスを入力してください:	あなたがしようとしている...
<input type="text"/>	mbed ユーザー アカウントを作成する
私は既にアカウントを持っています！	アカウントが mbed サイト やリソースにアクセスすること、あなたのためには設定されます。
ユーザー名を選択してください	
<input type="text"/>	
新しいパスワード	
<input type="text"/>	
パスワードの確認	
<input type="text"/>	
ファーストネーム:	
<input type="text"/>	
姓:	
<input type="text"/>	
Country:	
<input type="text"/> Select a country...	
<input type="checkbox"/> I agree to ARM's Terms and Conditions . (required)	
<input type="checkbox"/> I confirm I have read and accept ARM's Privacy Policy and indicate my consent to receiving marketing communications from ARM.	
Signup	

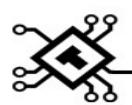




mbed登録(6)

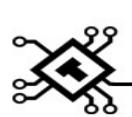
- うまく作れるとワークスペースが作られる。
- ファイルの管理などは、この画面で行なわれる。
- 登録したメールアドレスに登録確認のメールが届いているので、その中のURLをクリックして認証を済ませておく。





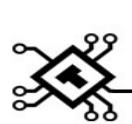
目次

- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- 8. LEDのスタティック点灯・ダイナミック点灯
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう



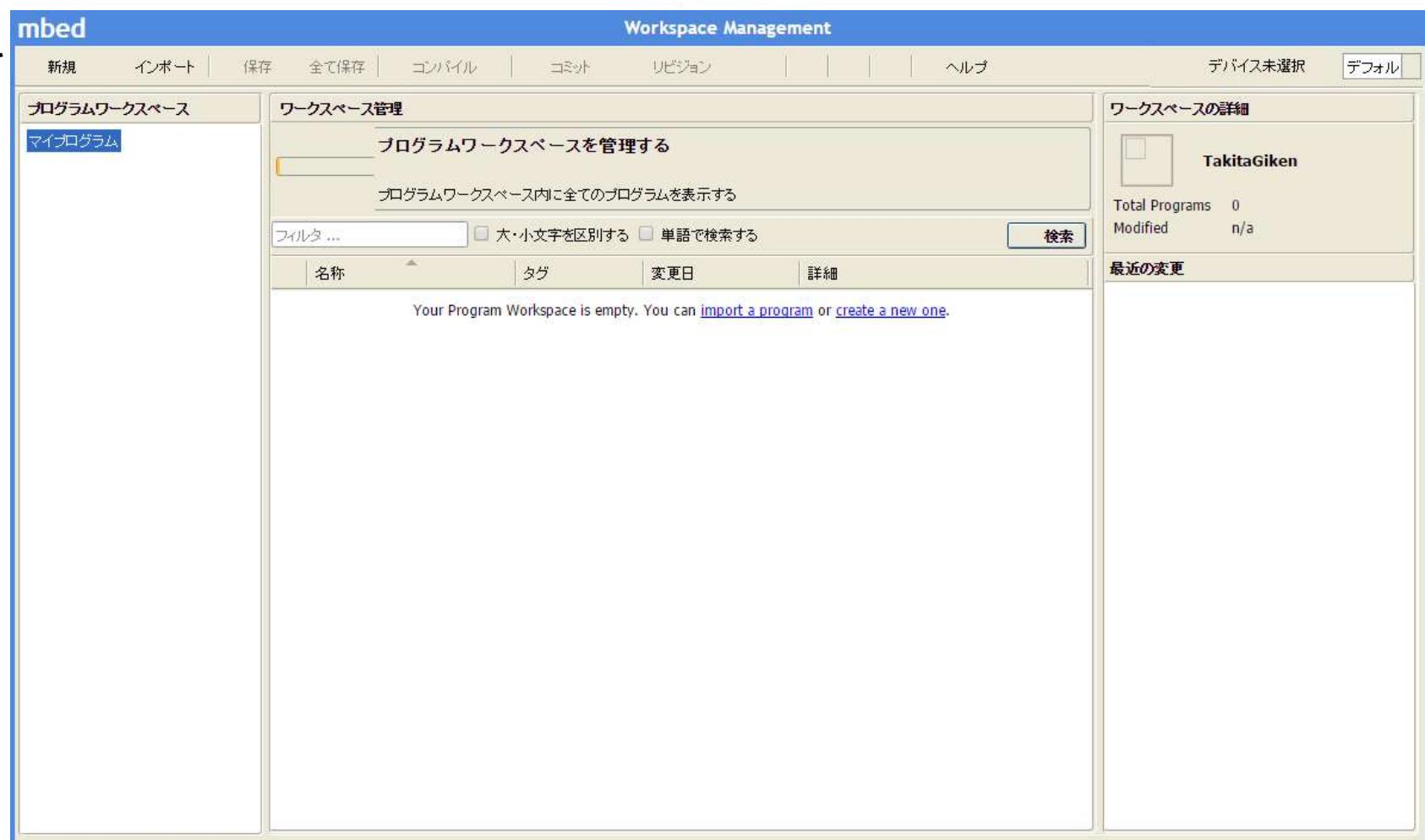
Lチカに挑戦(1)

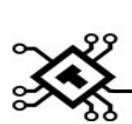
- ・バーサライタの基板に使用されているマイコンは, LPC1114FN28/102というマイコンである.
- ・このマイコンの初期プログラムをmbedを使用して開発する.



Lチカに挑戦(2)

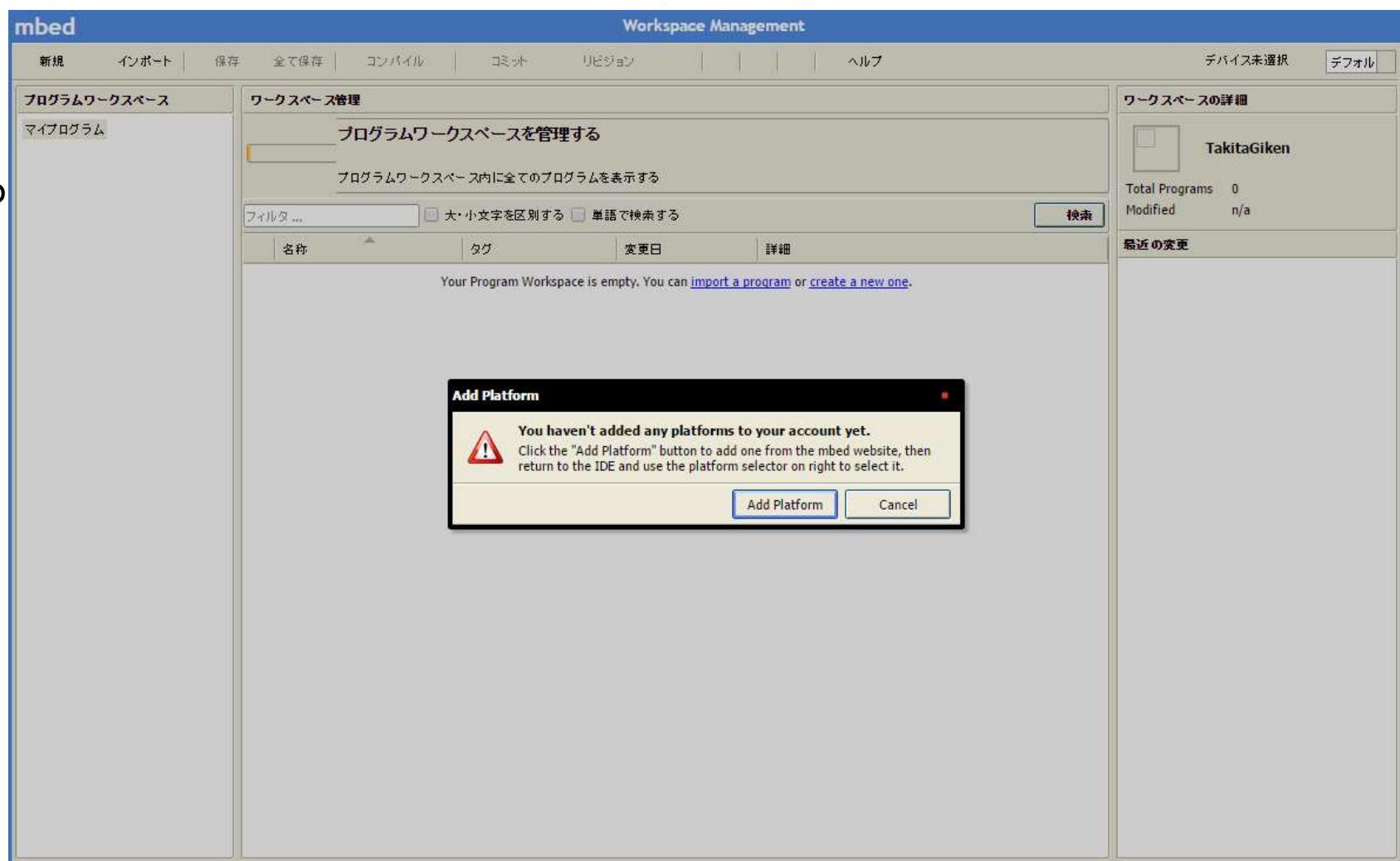
- 右図の新規作成を押すと、

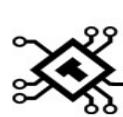




チカに挑戦(3)

- 右図の新規作成を押すと、プラットフォームを追加するように指示されるので **Add Platform** をおす。





Lチカに挑戦(4)

- 右図の新規作成を押すと、プラットフォームを追加するように指示されるので **Add Platform** をおす。
- 右図のようなプラットフォーム一覧が表示されるので、このなかの、

ARM mbed

Hardware Documentation Code Questions Forum TakitaGiken Compiler

Platforms

Search developer.mbed.org...

Filter

mbed Enabled

mbed Enabled

Target vendor

ARM
 Atmel
 Maxim Integrated
 NXP Semiconductors
 Nordic Semiconductor ASA
 Renesas
 STMicroelectronics
 Silicon Labs
 WIZnet

Platform vendor

ARM
 Atmel
 BBC Make it Digital Campaign
 CQ Publishing Co.,Ltd.
 Delta
 Embedded Artists
 Espotel
 JKSoft
 Maxim Integrated
 MultiTech
 NGX Technologies
 NXP Semiconductors
 Nordic Semiconductor ASA
 Outrageous Circuits
 RedBearLab
 Renesas
 STMicroelectronics
 SeeedStudio
 Semtech
 Silicon Labs
 Solder Splash Labs
 Switch Science Inc.
 WIZnet
 communitycontributors
 u-blox AG

Connectivity

Bluetooth Smart
 CAN
 Cellular
 Ethernet
 USB Device
 USB Host
 WiFi

Form Factor

Arduino Compatible
 Breadboardable
 XBee

Platforms

NXP

mbed LPC1768
• Cortex-M3, 96MHz
• 512KB Flash, 32KB RAM

NXP

mbed LPC11U24
• Cortex-M0, 48MHz
• 32KB Flash, 8KB RAM

NXP

FRDM-KL25Z
• Cortex-M0+
• 128KB Flash, 16KB RAM
• USB OTG

NXP

NXP LPC800-MAX
• Cortex-M0+
• 16KB Flash, 4KB RAM

NXP

EA LPC4088 QuickStart Board
• Cortex-M4, 120MHz
• 512KB Flash, 96KB SRAM

NXP

DipCortex M0
• Cortex-M0, 50MHz
• 32KB Flash, 8KB RAM

NXP

DipCortex M3
• Cortex-M3, 72MHz
• 64KB Flash, 12KB RAM

NGX

BlueBoard-LPC11U24
• Cortex-M0, 48MHz
• 32KB Flash, 8KB RAM

NXP

WiFi DipCortex
• Cortex-M3, 72MHz
• 64KB Flash, 12KB RAM

NXP

Seeeduino-Arch
• Cortex-M0, 48MHz
• 32KB Flash, 8KB RAM

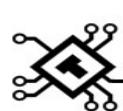
NXP

mbed LPC1114FN28
• Cortex-M0, 48MHz
• 32KB Flash, 4KB RAM

u-blox

u-blox C027
• Cortex-M3, 96MHz
• 512 KB Flash, 32KB RAM
• Onboard cellular module

Nippon Institute of Technology



Lチカに挑戦(5)

- 右図の新規作成を押すと、プラットフォームを追加するように指示されるので **Add Platform** をおす。
- 右図のようなプラットフォーム一覧が表示される
- mbed LPC1114FN28を選らぶ。

ARM mbed

Hardware Documentation Code Questions Forum TakitaGiken Compiler

Platforms

Search developer.mbed.org...

Filter

mbed Enabled

mbed Enabled

Target vendor

ARM
 Atmel
 Maxim Integrated
 NXP Semiconductors
 Nordic Semiconductor ASA
 Renesas
 STMicroelectronics
 Silicon Labs
 WIZnet

Platform vendor

ARM
 Atmel
 BBC Make it Digital Campaign
 CQ Publishing Co.,Ltd.
 Delta
 Embedded Artists
 Espotel
 JKSoft
 Maxim Integrated
 MultiTech
 NGX Technologies
 NXP Semiconductors
 Nordic Semiconductor ASA
 Outrageous Circuits
 RedBearLab
 Renesas
 STMicroelectronics
 SeeedStudio
 Semtech
 Silicon Labs
 Solder Splash Labs
 Switch Science Inc.
 WIZnet
 communitycontributors
 u-blox AG

Connectivity

Bluetooth Smart
 CAN
 Cellular
 Ethernet
 USB Device
 USB Host
 WiFi

Form Factor

Arduino Compatible
 Breadboardable
 XBee

Platforms

NXP

mbed LPC1768
• Cortex-M3, 96MHz
• 512KB Flash, 32KB RAM

NXP

mbed LPC11U24
• Cortex-M0, 48MHz
• 32KB Flash, 8KB RAM

NXP

FRDM-KL25Z
• Cortex-M0+
• 128KB Flash, 16KB RAM
• USB OTG

NXP

NXP LPC800-MAX
• Cortex-M0+
• 16KB Flash, 4KB RAM

NXP

EA LPC4088 QuickStart Board
• Cortex-M4, 120MHz
• 512KB Flash, 96KB SRAM

NXP

DipCortex M0
• Cortex-M0, 50MHz
• 32KB Flash, 8KB RAM

NXP

DipCortex M3
• Cortex-M3, 72MHz
• 64KB Flash, 12KB RAM

NGX

BlueBoard-LPC11U24
• Cortex-M0, 48MHz
• 32KB Flash, 8KB RAM

NXP

WiFi DipCortex
• Cortex-M3, 72MHz
• 64KB Flash, 12KB RAM

NXP

Seeeduino-Arch
• Cortex-M0, 48MHz
• 32KB Flash, 8KB RAM

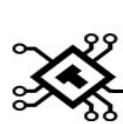
NXP

mbed LPC1114FN28
• Cortex-M0, 48MHz
• 32KB Flash, 4KB RAM

NXP

u-blox C027
Cortex-M3, 96MHz
512 KB Flash, 32KB RAM
Onboard cellular module

Nippon Institute of Technology



Lチカに挑戦(6)

- 右図の新規作成を押すと、プラットフォームを追加するように指示されるので **Add Platform** をおす。
- 右図のようなプラットフォーム一覧が表示される
- mbed LPC1114FN28を選らぶ。
- mbed LPC1114FN28の詳細が書かれたページに飛ぶ。
- Add to your mbed Compiler** を押すと、**Open mbed Compiler** に切り替わり、登録が完了する。

ARMmbed

Hardware Documentation Code Questions Forum TakitaGiken Compiler

Platforms » mbed LPC1114FN28

Search developer.mbed.org... Search

Platform Partner

SWITCHSCIENCE

Switch Science
Switch Science is one of the open source hardware retailers in Japan. We are manufacturing our own products.

Silicon Partner

NXP

NXP
NXP is a leading semiconductor company founded by Philips more than 50 years ago.

Open mbed Compiler

Remove from your mbed Compiler
You have this platform in your mbed Compiler, if you do not use it you can remove it.

Remove Follow

Platform 'mbed LPC1114FN28' is now added to your account!

mbed LPC1114FN28

The LPC1114FN28 is an ARM Cortex-M0 based, low-cost 32-bit MCU, designed for 8/16-bit microcontroller applications, offering performance, low power, simple instruction set and memory addressing together with reduced code size compared to existing 8/16-bit architectures.



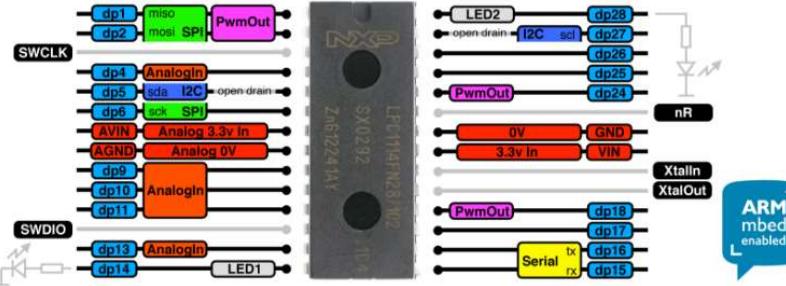
ARM mbed enabled

Table of Contents

- Pinout
- Available Packages
- Data Sheets
- See also
- Firmware
- Credits

Note: LPC1114FN28 platform doesn't support RTOS due to its flash size. Please do not import mbed-rtos library into your project.

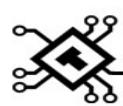
Pinout



Pinout diagram showing the connections for the mbed LPC1114FN28. The diagram includes the following connections:

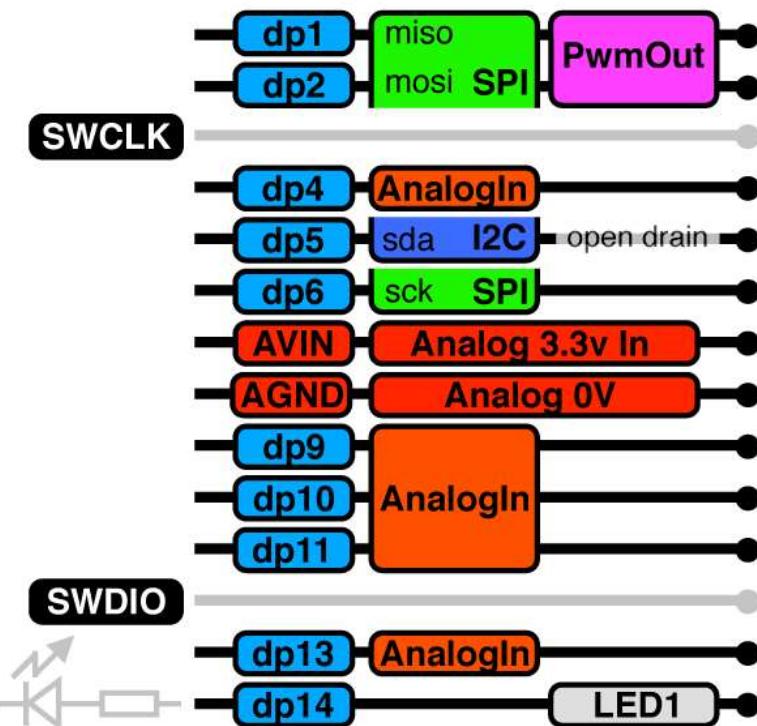
- SWCLK: dp4 (mosi), dp2 (miso), SPI
- SWDIO: dp13 (AnalogIn), dp14 (LED1)
- AVIN: dp6 (sda), dp5 (sck), SPI
- AGND: dp9 (Analog 3.3v In), dp10 (Analog 0V)
- LED2: dp23 (PwmOut), dp27 (open drain), dp28 (I2C scl), dp29 (PwmOut), dp24 (nR), dp25 (0V), dp26 (GND), dp20 (3.3v In), dp19 (VIN)
- Serial: dp18 (PwmOut), dp17 (Serial tx), dp16 (Serial rx), dp15 (XtalIn), dp14 (XtalOut)

ARM mbed enabled



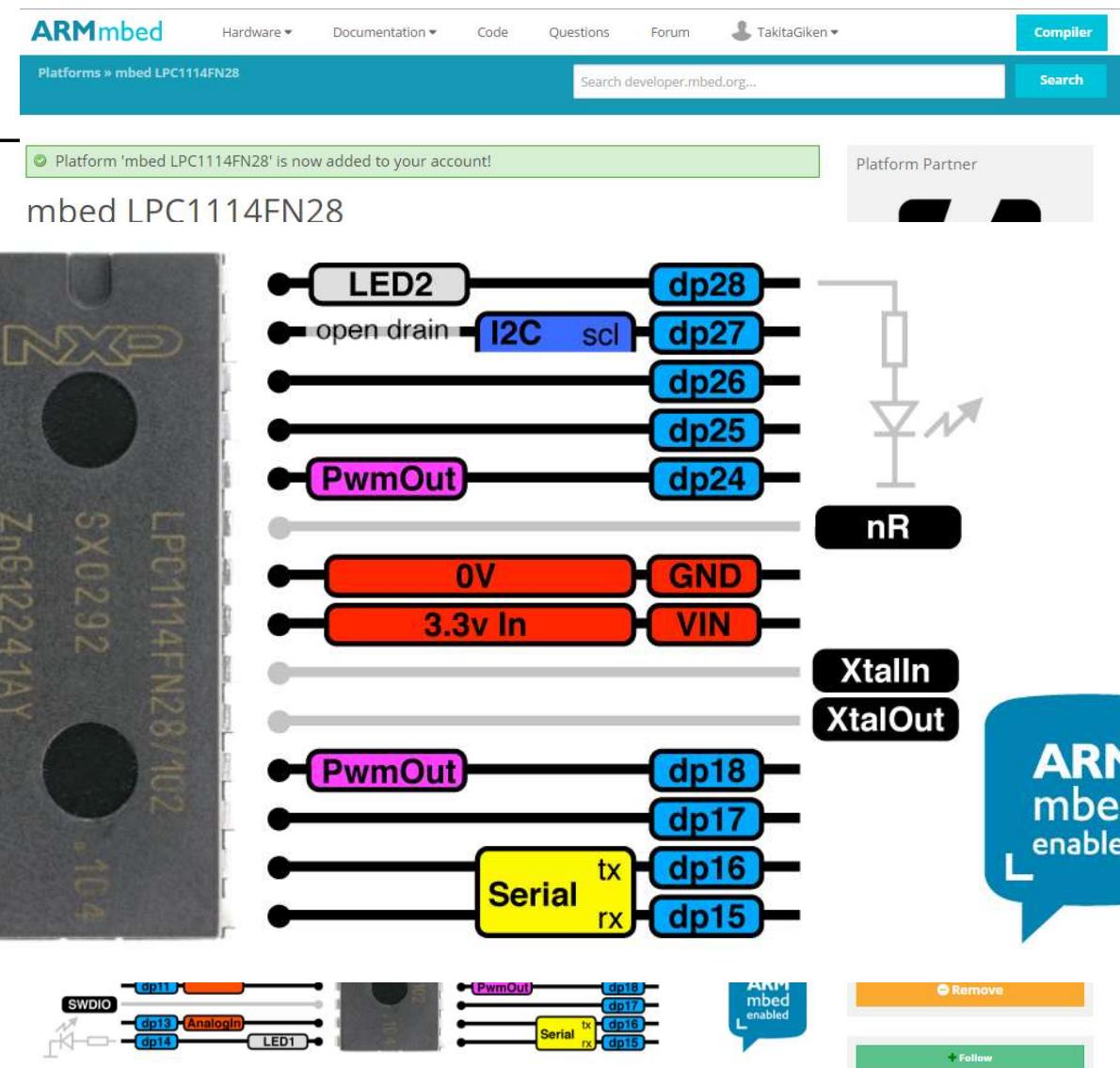
Lチカに挑戦(7)

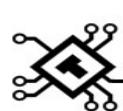
- 右図の新規作成を



すと、
に切り替わり、登録
が完了する。

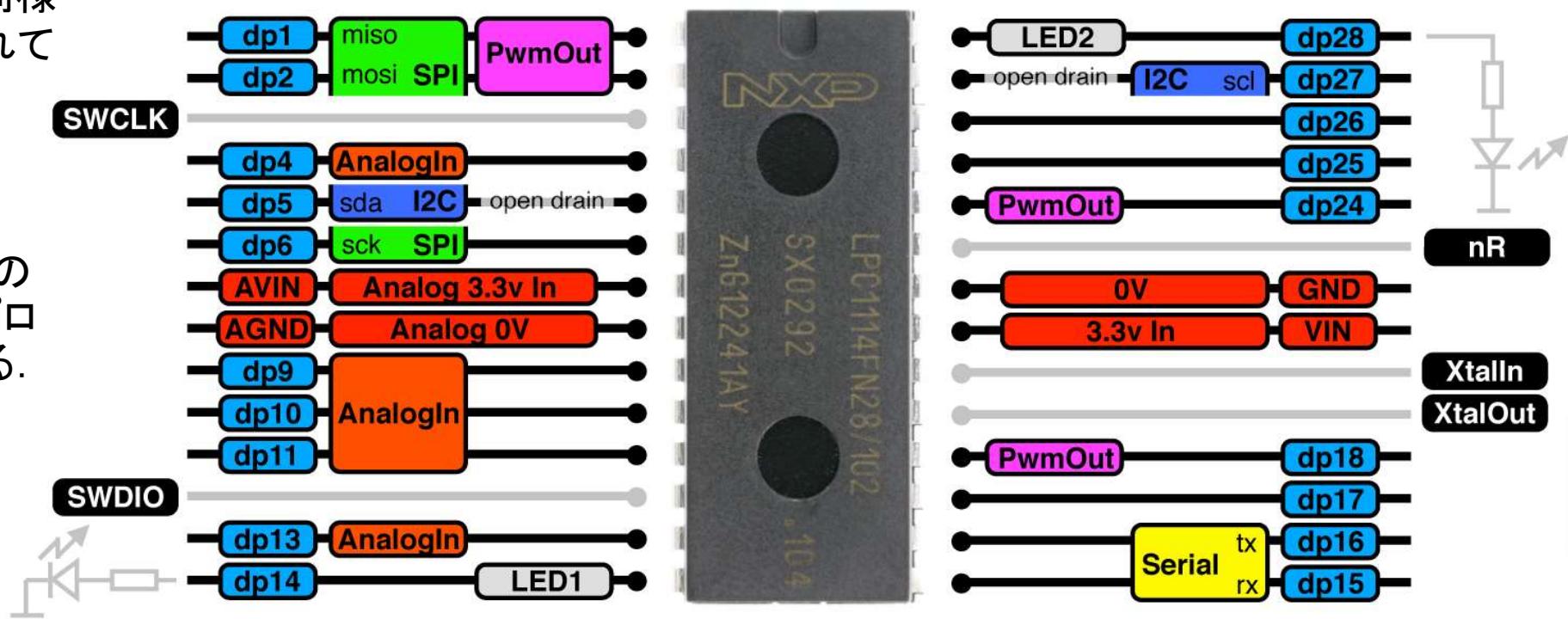
TAKITA Lab.

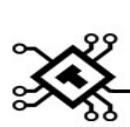




Lチカに挑戦(9)mbed LPC1114FN28

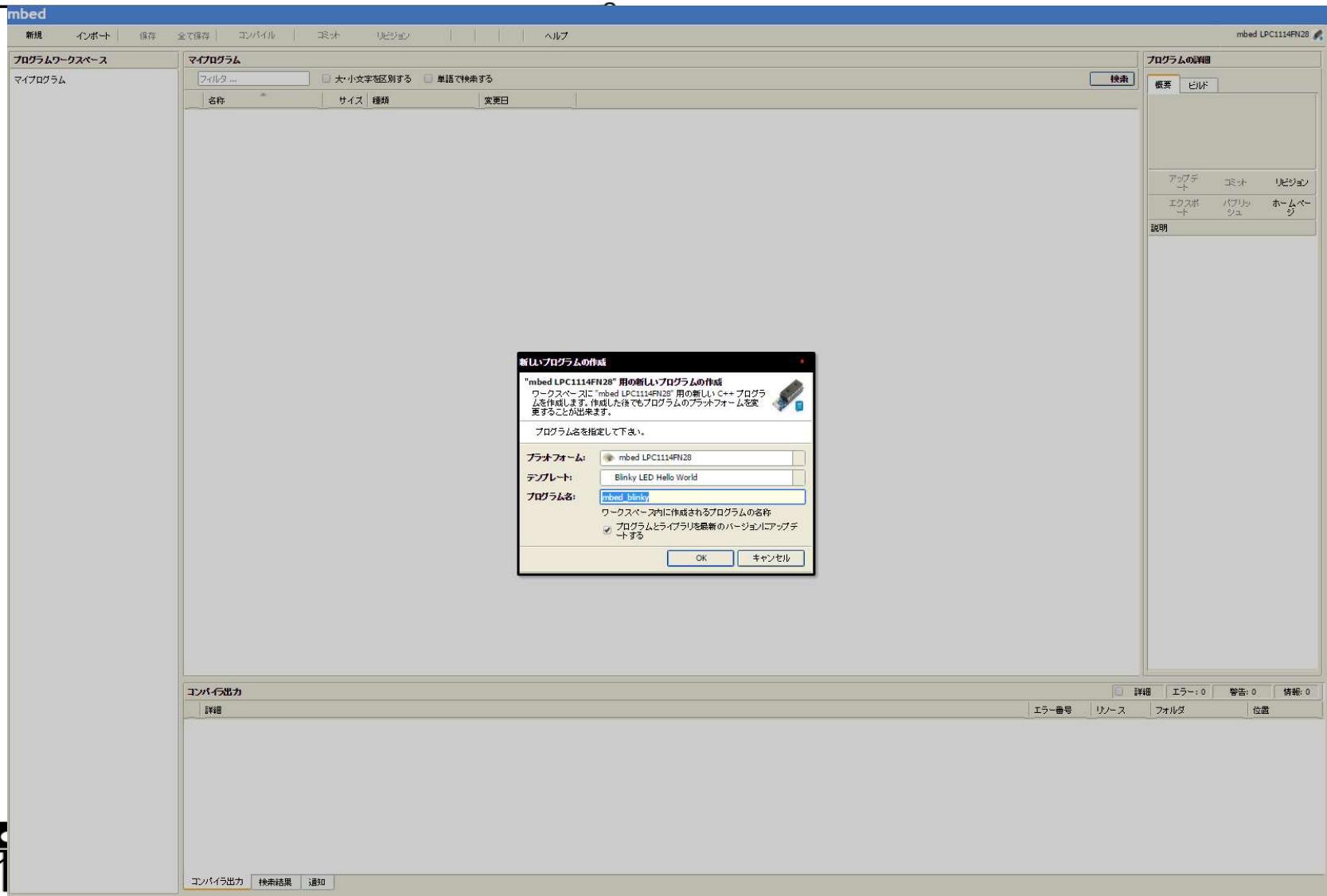
- LEDAは、mbed LPC1114FN28と同様のピンに接続されている。
- 28ピン、
- 14ピン
- まずは、この2つのLEDを点灯するプログラムを作成する。

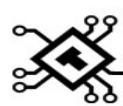




「チカに挑戦(10)

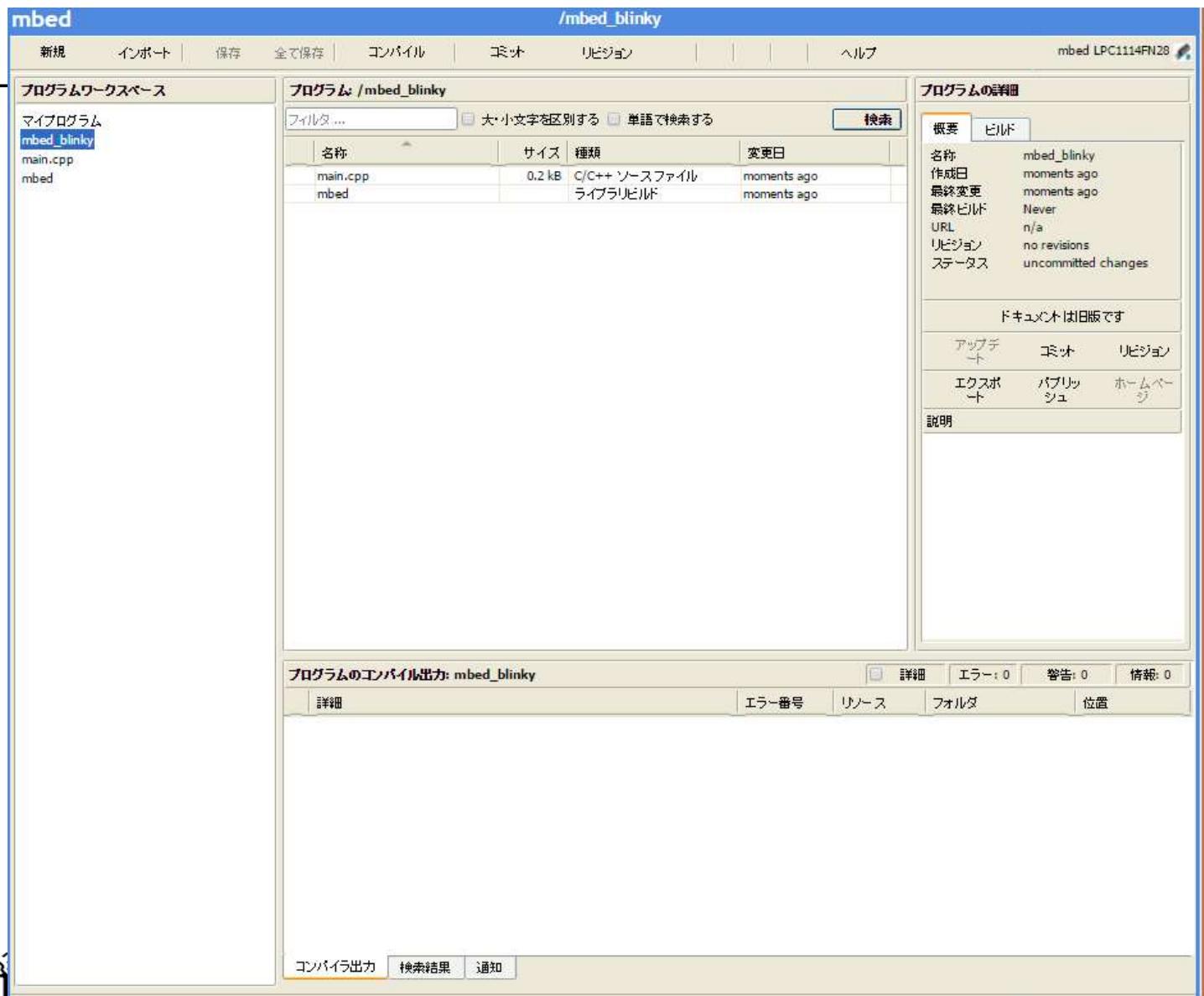
- ・ **Open mbed Compiler** をおすと、右図が表示される。
- ・ プラットフォームが mbed LPC1114FN28 になっていることを確認する。
- ・ プログラム名は “mbed_blinky” のままでよい。
- ・ [ok]をおすと、

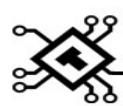




Lチカに挑戦(11)

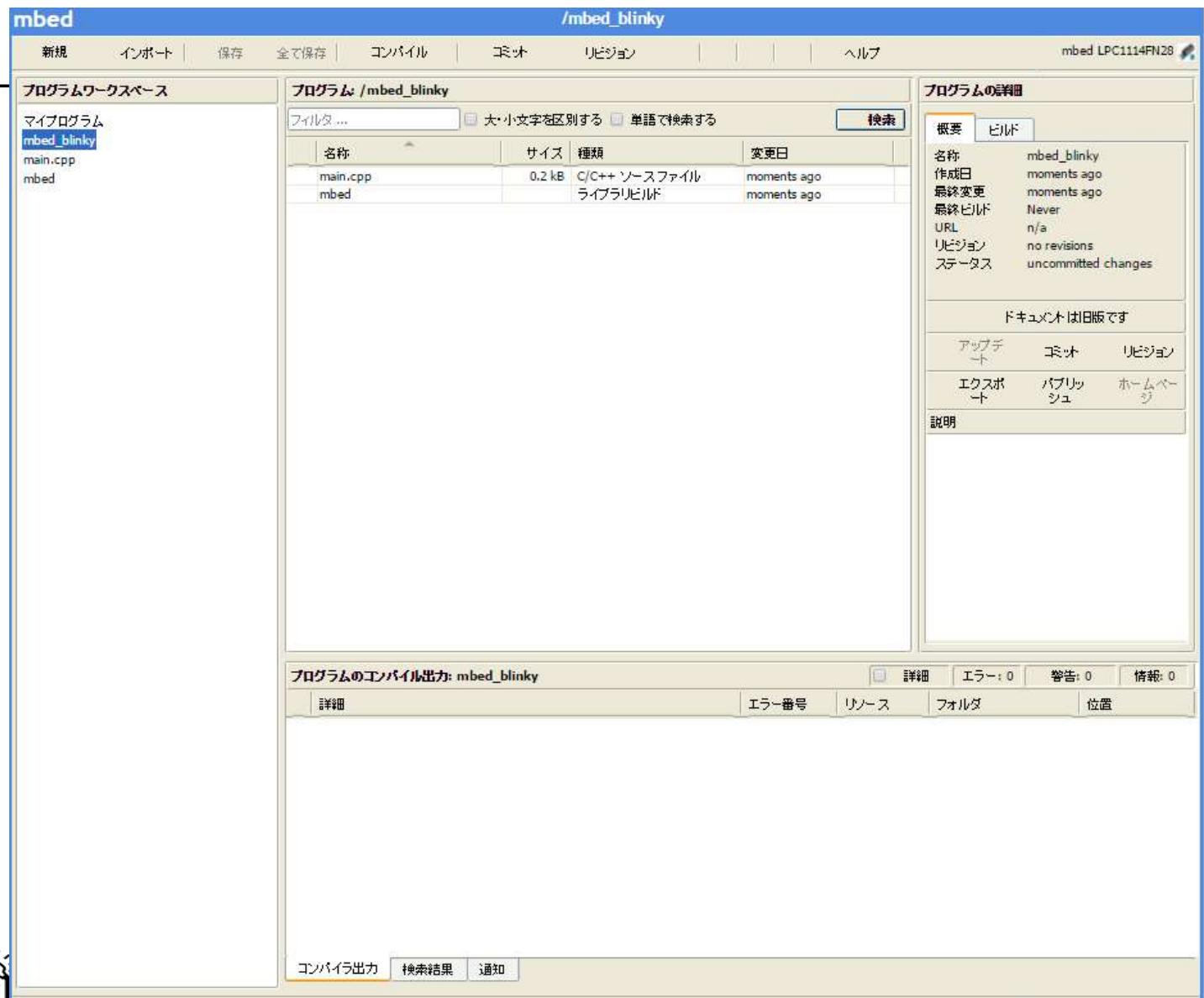
- **Open mbed Compiler** をおすと、右図が表示される。
- プラットフォームが mbed LPC1114FN28 になっていることを確認する。
- プログラム名は “mbed_blinky”のままでよい。
- [ok]をおすと、mbed_blinkyプロジェクトが作られる。

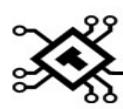




Lチカに挑戦(12)

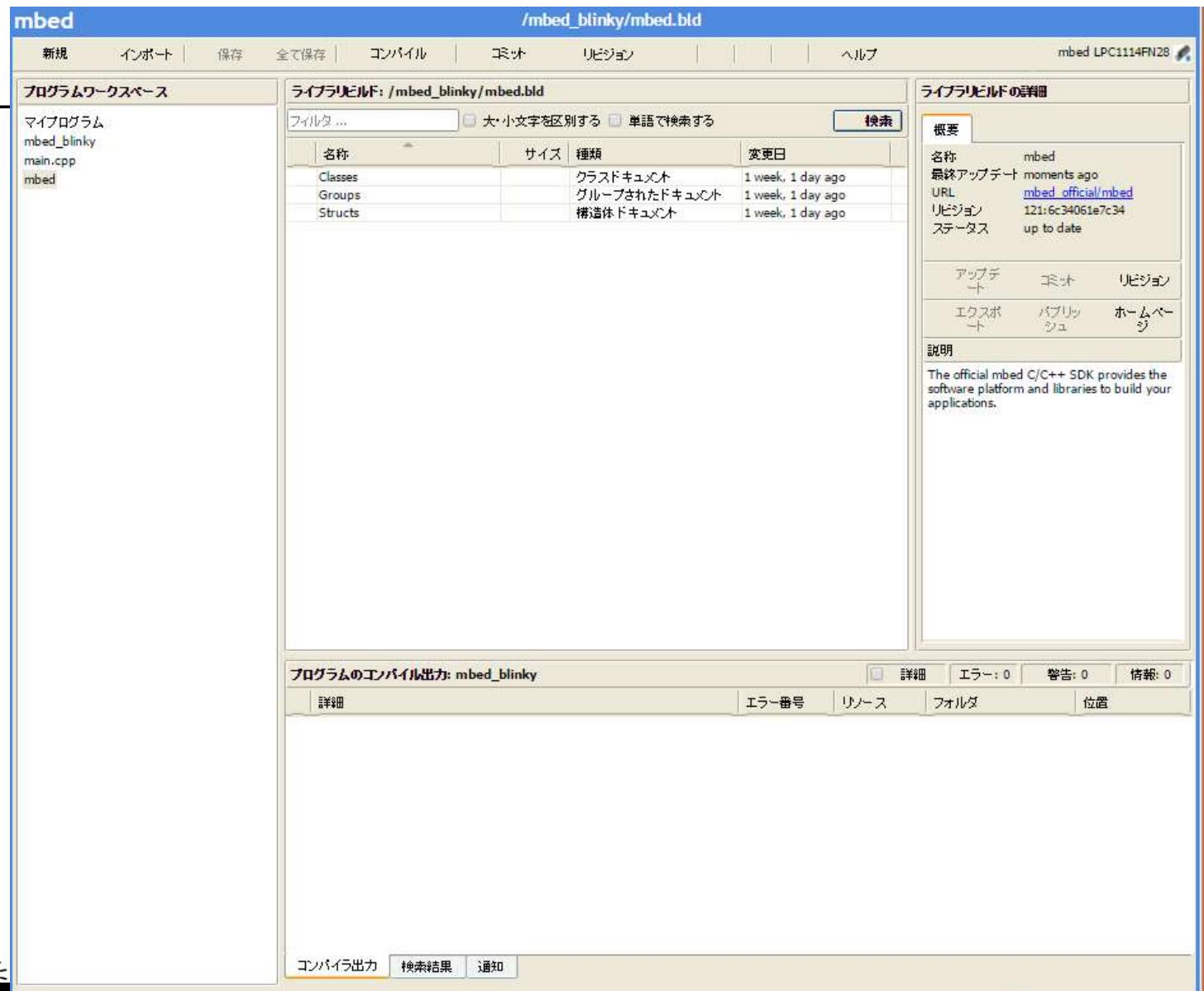
- **Open mbed Compiler** をおすと、右図が表示される。
- プラットフォームが mbed LPC1114FN28 になっていることを確認する。
- プログラム名は “mbed_blinky”のままでよい。
- [ok]をおすと、mbed_blinkyプロジェクトが作られる。
- main.cppがプログラムの本体となるファイルである。

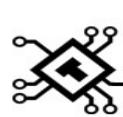




チカに挑戦(13)

- プログラムワークスペースの「mbed」をクリックすると, Classes, Groups, Structsが表示される.
- Classesを開くと, 選んだプラットフォームで使える class(機能)が表示される.
- Structsでは, 予め定義されている構造体が表示される.
- Groupsはグループ化された関数のドキュメントが入っている.

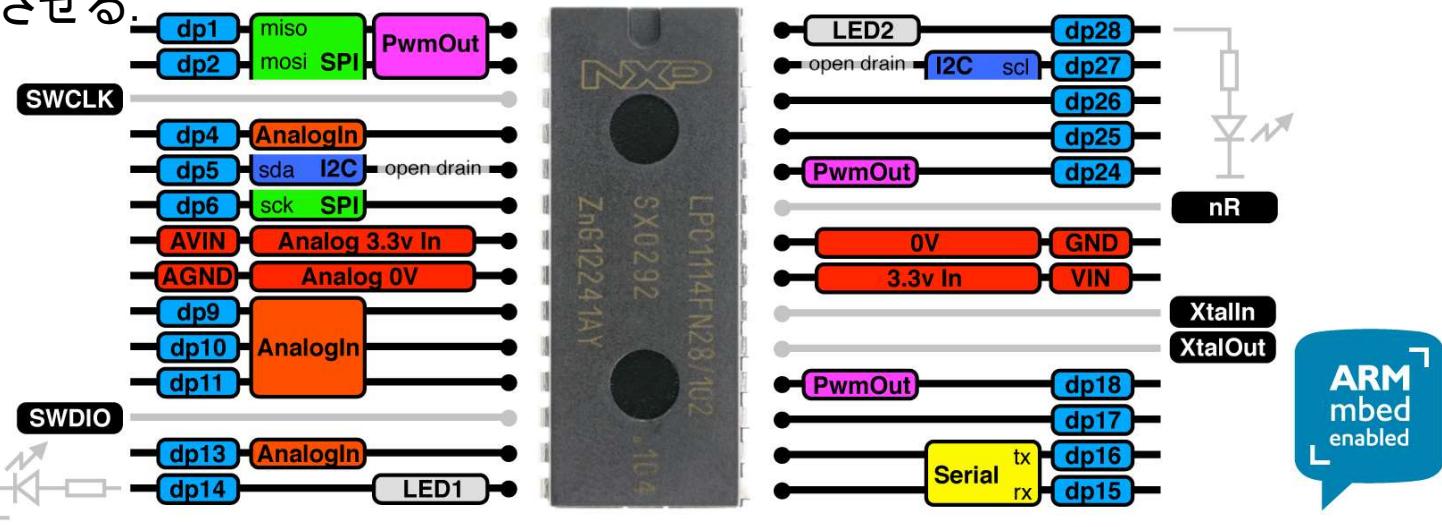


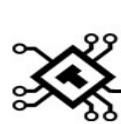


Lチカに挑戦(14)

- main.cppをクリックするとプログラムのソースコードが表示される。
- DigitalOutは、デジタル出力の型名である。同時にclassの一つでもある。
- myledが使用する変数名、(LED1)は、myledにLED1(14番ピン)をわり当てようとしている。
- while(1)で無限ループを作る。
- myledに1/0を0.2秒毎に、変化させる。

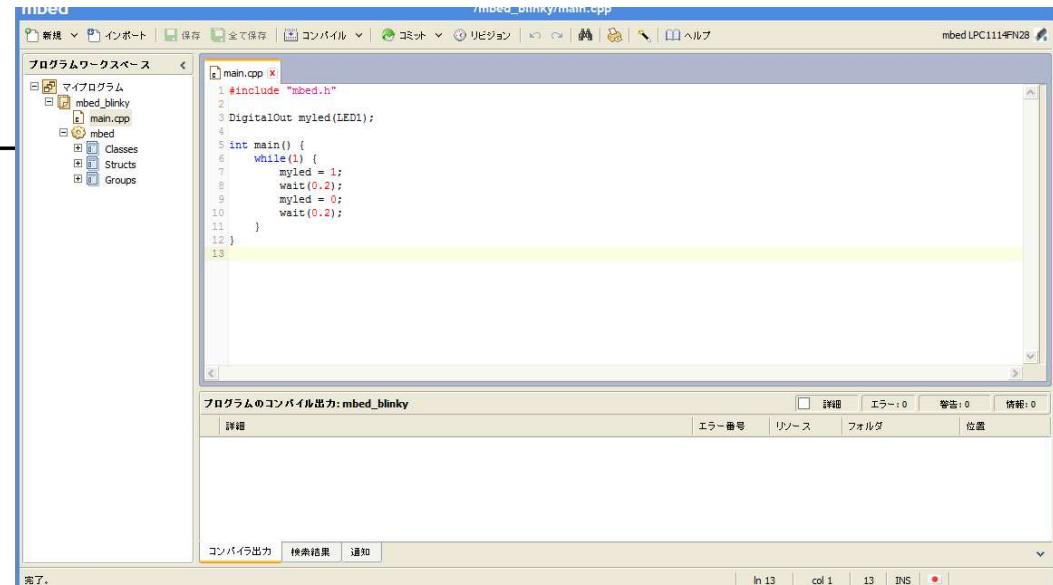
```
main.cpp AsynchI2C
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
13
```





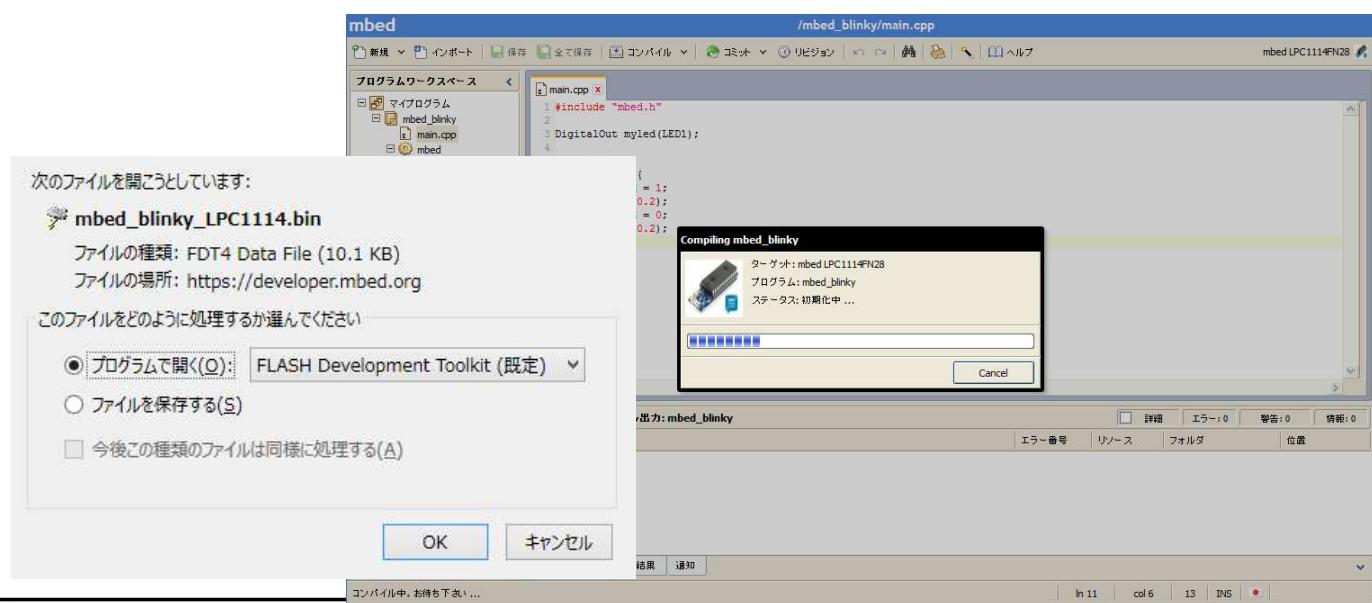
Lチカに挑戦(15)

- すべて保存し、コンパイルボタンと押すと、コンパイルが実行されて、実行ファイル(ファームウェアのバイナリ形式ファイル)がダウンロードされてくる。



```
#include "mbed.h"
DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```



次のファイルを開こうとしています:

mbed_blinky_LPC1114.bin

ファイルの種類: FDT4 Data File (10.1 KB)
ファイルの場所: <https://developer.mbed.org>

このファイルをどのように処理するか選んでください

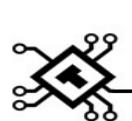
プログラムで開く(O): **FLASH Development Toolkit (既定)**

ファイルを保存する(S)

今後この種類のファイルは同様に処理する(A)

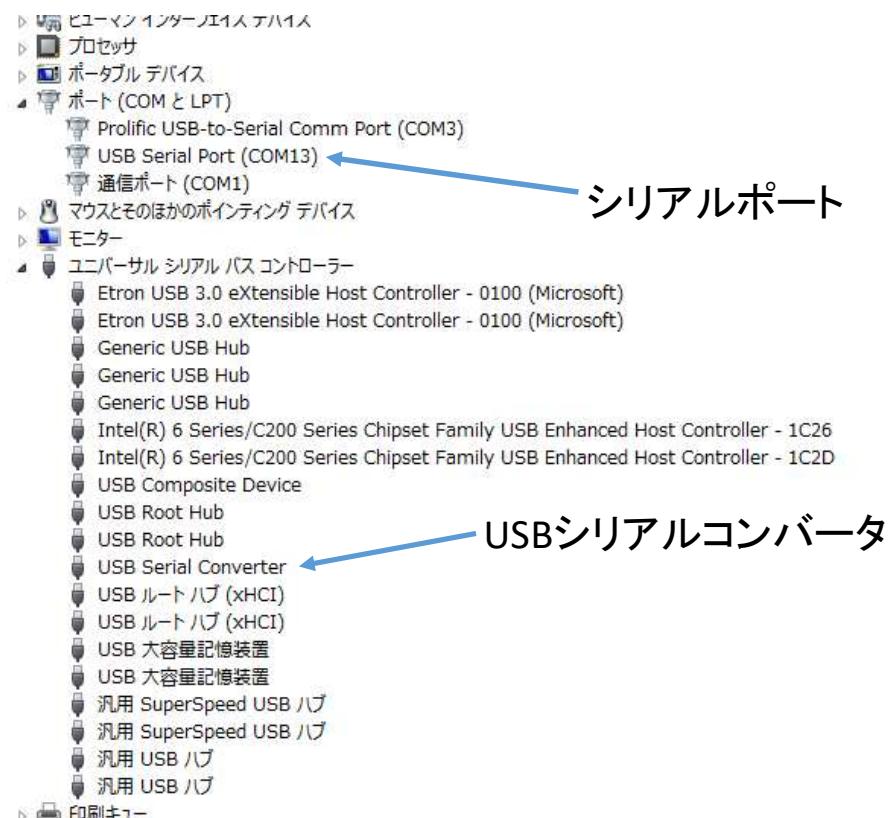
OK **キャンセル**

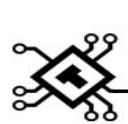
コンパイル中。お待ち下さい...



「チカに挑戦(16)

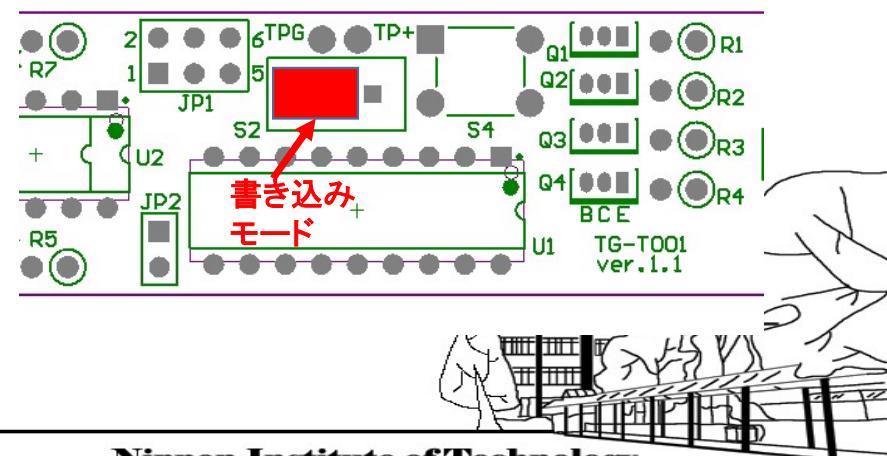
- ・バーサライタ基板には、USBシリアル変換アダプタを接続する(向きに注意)
- ・変換チップのドライバは、<http://www.ftdichip.com/Drivers/VCP.htm> にあるので、OSに合致するものをダウンロードする。
- ・ドライバは2種類インストールする。
- ・一つは、USB-シリアルコンバータ自体のドライバと、もう一つは、シリアルポートのドライバである。
- ・上記のWEBページからWindowsの場合、 “[展開したフォルダ]¥CDM v2.12.16 WHQL Certified” に ftdibus.inf, ftdiport.inf の二つが作られる。この二つがUSBシリアルコンバータのドライバとシリアルポートのドライバである。
- ・両方をインストールしたら、デバイスマネージャなどで、接続したUSBシリアルポートのCOM番号を確認しておく。

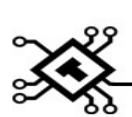




Lチカに挑戦(17)LPCISP

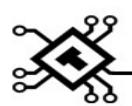
- 電源スイッチをOFFにして、電池・電源からの電力の供給を止める
- バーサライタ基板をPCと接続し、S2を書き込みモードの位置(S2の文字のある側)にして、リセットボタンS1を押す。これでプログラムモードに切り替わる。
- LPCISPを起動する。
- binファイルには、ダウンロードした mbed_blinky_LPC1114.binを指定する。
- COM番号は、Windowsのデバイスマネージャ等で確認しておく。
- 「書き込む」ボタンを押すと書き込みが行なわれ、終了したらUSBケーブルを抜き、電源スイッチをONにする。
- S2を実行モードの位置(S2の文字の反対側)にして、リセットボタンS1を押すとDS18が点滅する。





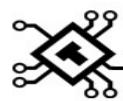
Lチカに挑戦(18) (課題)

- LEDは、DS17, DS18の二つが基板上に搭載されている。
- その二つを交互に点灯するプログラムを作成せよ。



目次

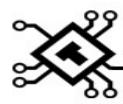
- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- 8. LEDのスタティック点灯・ダイナミック点灯
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう



下準備(1)

- ・プログラムを作っていると間違いや勘違いでプログラムが意図したとおりに動かないことがある。そのような時には、処理を追跡するための手段が必要である。また、プログラムによっては、変数の状態を確認することが有用であることもおおい。
- ・バーサライタ基板では、シリアルポートがあるので、シリアルポートを使用して、printfデバッグをする。
- ・デバッグ方法には、他に、JTAGデバッグやSWDデバッグがあるが、バーサライタ基板で利用できるのは、printf デバッグと SWDデバッグ(別途装置が必要)である。
- ・mbedでは、Serialクラスを使うことで容易にシリアルポートを使用することが可能である。
- ・Serial pc(dp16,dp15) でシリアルポートを設定する。
- ・9行目のpc.baud(115200)で通信速度を設定している。
- ・15行目のpc.printfで文字を表示する。ここでは、tickの数字を表示している。

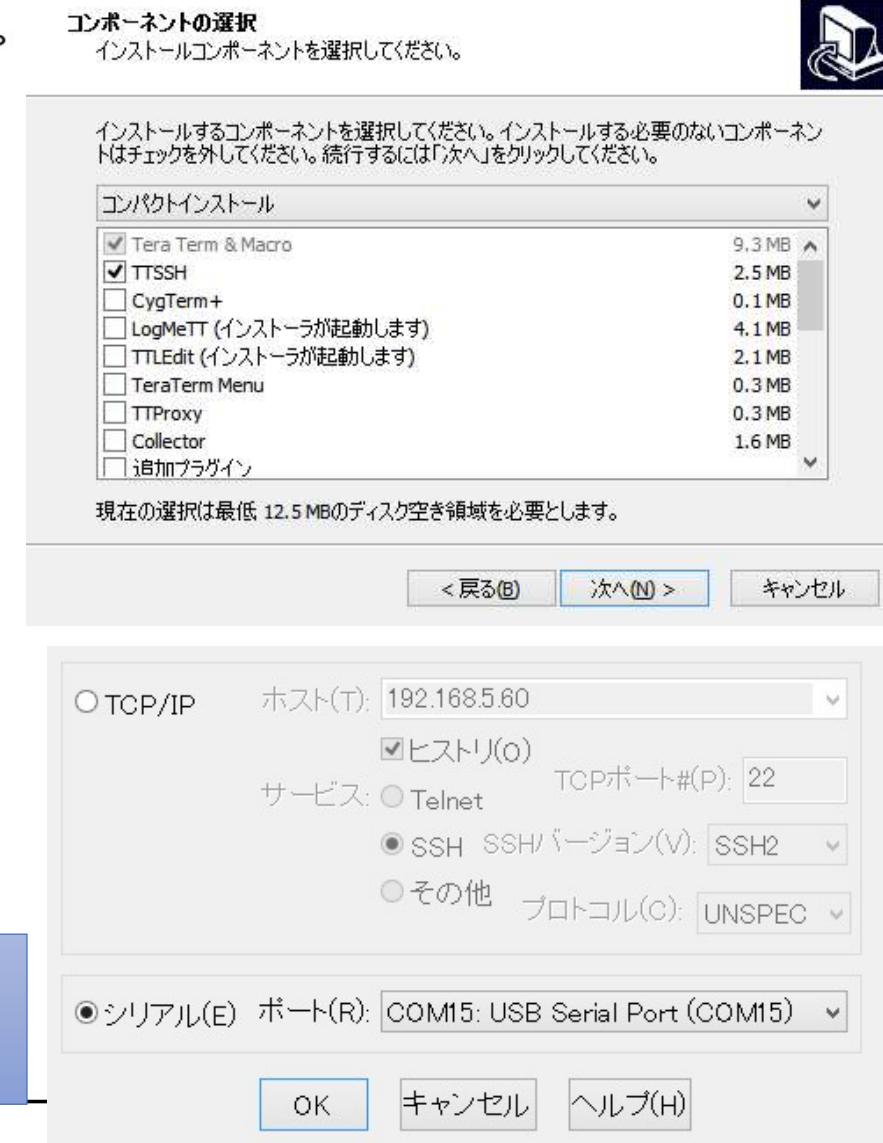
```
1. #include "mbed.h"
2.
3. DigitalOut myled(LED1);
4. Serial pc(dp16,dp15);
5.
6. int main() {
7.     uint32_t tick=0 ;
8.
9.     pc.baud(115200);
10.    while(1) {
11.        myled = 1;
12.        wait(0.2);
13.        myled = 0;
14.        wait(0.2);
15.        pc.printf("tick = %d \n\r",tick++);
16.    }
17. }
```

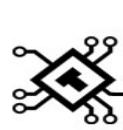


下準備(2)

- PC側にターミナルアプリがないとシリアルポートからきた情報を表示できないので、ここでは、teratermを使用する。
- <https://ttssh2.osdn.jp/> からダウンロードページに移行し、Teratermをダウンロードする。
- 執筆時点での最新版は、[teraterm-4.91.exe](#)である。これをダウンロードして、実行する。
- インストーラが立ち上がるるので、その指示に従ってインストールをする。
- コンポーネントの選択では、シリアルコンソールを使用するだけならば、「コンパクトインストール」で十分である。
- インストール後に、実行し、接続先にシリアルを選択し、速度を115200に設定すると、tickが表示される。

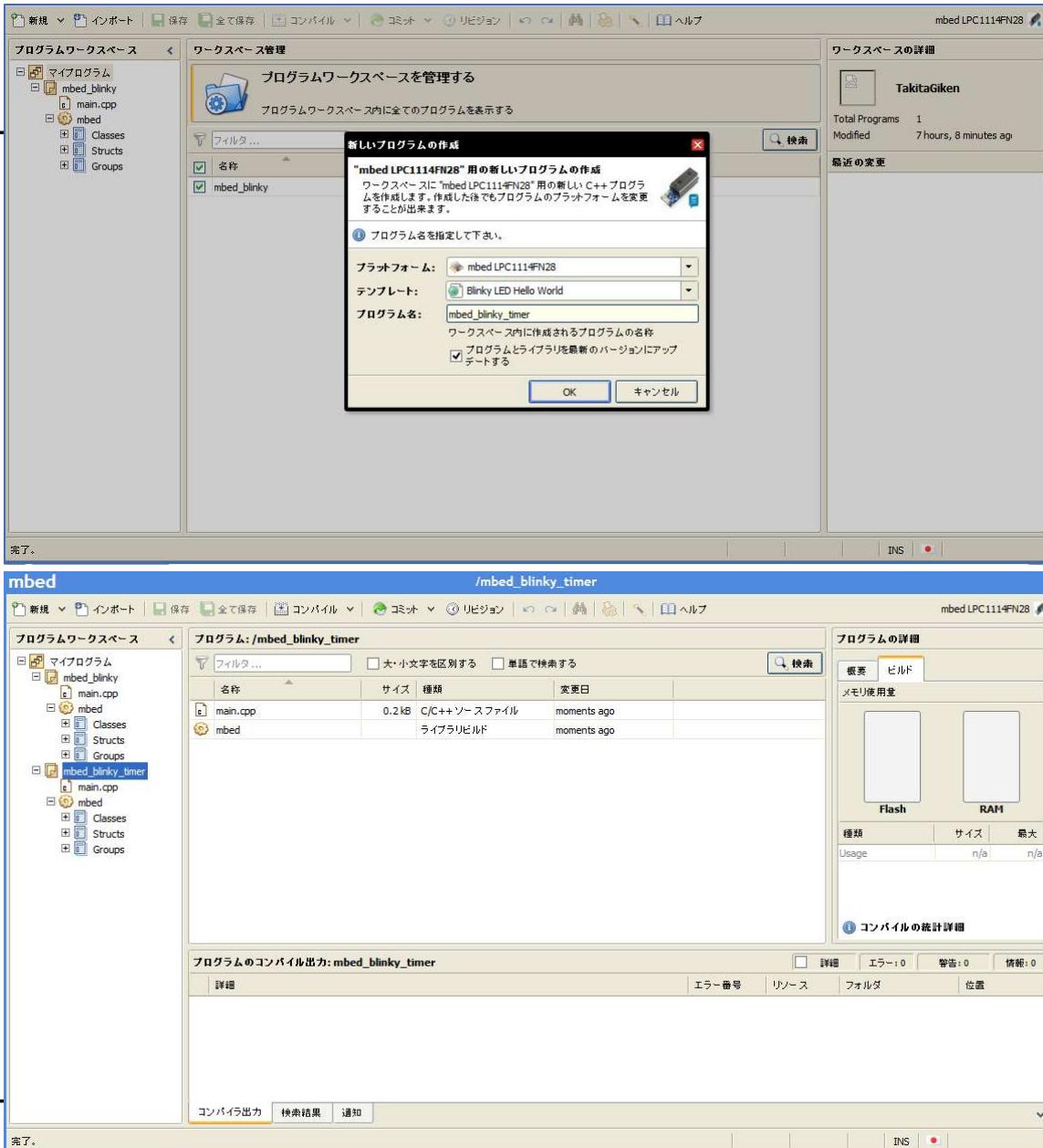
```
tick = 34
tick = 35
tick = 36
```

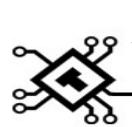




タイマをつかう

- 新規にプロジェクトを立ち上げる。
- メニューの新規を押すと、新しいプログラムの作成のダイアログが出ている。
- プログラム名を適当な名前（ここでは、「mbed_blinky_timer」としておく）。
- OKを押すと、ワークスペースに mbed_blinky_timer が追加される。
- main.cppを開き、タイマーを使った処理を追加する。





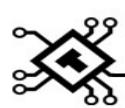
タイマをつかう

- mbed LPC1114FN28では、タイマを使ったClassがいくつか定義されているが、ここでは、2つのタイマについて説明する。
- 一つは、Timerクラス。これは、汎用タイマーで、開始してからの時間を計るのに使われる。
- もう一つは、Tickerクラス。これは、一定時間毎に繰り返し発生する処理をするために使われる。



Timer

- Timerクラスを使用して、実行してからの時間を表示させることを考える。
- Timerを使ったプログラム開発のために新しいプログラムを作成する。
- mbedのワークスペースに新規にプログラム「mbed_blinky_timer」を追加する。
- Timer tmr を定義する。
- Timerクラスが持つ関数は以下の通り。
 - tmr.start()でタイマを開始
 - tmr.stop()でタイマを停止
 - tmr.reset()でタイマの初期化
 - tmr.read()でタイマの現在値を取得
- これらの関数を使って時間を計る。



Timer

- TimerとSerialを追加したプログラムをコンパイルし, 得られたbinファイルをマイコンにダウンロードする.
- 実行すると, 浮動小数点で, 実行時間が表示される.
- 時間の間隔は, 0.414689前後でほぼ一定である.

```
1. #include "mbed.h"
2.
3. DigitalOut myled(LED1);
4. Serial pc(dp16,dp15);
5. Timer tmr;
6.
7. int main() {
8.     pc.baud(115200);
9.     tmr.start();
10.    while(1) {
11.        myled = 1;
12.        wait(0.2);
13.        myled = 0;
14.        wait(0.2);
15.        pc.printf("time: %lf\n",tmr.read());
16.    }
17. }
```

```
time: 0.400013
time: 0.8 14706
time: 1.229395
time: 1.644081
time: 2.058770
time: 2.473456
```

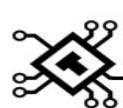


Timer

- TimerとSerialを追加したプログラムをコンパイルし, 得られたbinファイルをマイコンにダウンロードする.
- 実行すると, 浮動小数点で, 実行時間が表示される.
- 時間の間隔は, 0.414689前後でほぼ一定である.
- この時間は, wait(0.2)が2個あることが支配的な要因である.
- しかし, whileのなかで, 処理が増えてくると, この時間を一定に保つのが難しい.
- 例えば, 外部から一定ではない間隔でやってくる信号がある場合を考慮する.
- 14行目のwait()の引数を, rand()で生成すると, 間隔は一定でなくなる.

```
1. #include "mbed.h"
2.
3. DigitalOut myled(LED1);
4. Serial pc(dp16,dp15);
5. Timer tmr;
6.
7. int main() {
8.     pc.baud(115200);
9.     tmr.start();
10.    while(1) {
11.        myled = 1;
12.        wait(0.2);
13.        myled = 0;
14.        wait((float)rand() / RAND_MAX);
15.        pc.printf("time: %lf\n", tmr.read());
16.    }
17. }
```

```
time: 0.200018
time: 1.242300
time: 2.109421
time: 3.161578
time: 3.429670
time: 4.402697
```

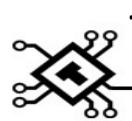


Ticker

- Tickerクラスは定期的に関数を実行するクラスである。
- attach_us(30行目)またはattach(31行目)で定期実行(100us毎, 0.2sec毎)する関数を定義している。
- 9~12行は、100us毎にtickを1増やして、基準となる時間を記録している。
- 14~25行でLEDの点滅と時間の表示を行なっている。
- 33行からのwhileループでは、乱数で発生させた時間だけwaitがかかるが、定期実行されている関数の起動には、影響していない。

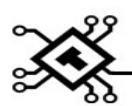
```
time: 0.200000
time: 0.600000
time: 1.000000
time: 1.400000
time: 1.800000
time: 2.200000
```

```
1. #include "mbed.h"
2.
3. DigitalOut myled(LED1);
4. Serial pc(dp16,dp15);
5. Ticker tick_handler;
6. Ticker serial_handler;
7. static uint64_t tick;
8.
9. void tick_int(void)
10. {
11.     tick++;
12. }
13.
14. void serial_int(void)
15. {
16.     static uint8_t led_on = 0;
17.     if(led_on==0){
18.         myled = 0;
19.         led_on=1;
20.         pc.printf("time: %f\n", (double)tick/10000.0);
21.     }else{
22.         myled = 1;
23.         led_on=0;
24.     }
25. }
26.
27. int main()
28. {
29.     pc.baud(115200);
30.     tick_handler.attach_us(&tick_int,100);
31.     serial_handler.attach(&serial_int,0.2);
32.
33.     while(1) {
34.         wait((float)rand()/RAND_MAX);
35.     }
36. }
```



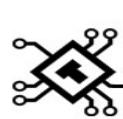
Ticker(課題)

- 前述のリストを改造して, main関数中のwhileループで, 時間を([時]:[分]:[秒])で表示するプログラムを作成せよ.
- なお, tickは, 100us毎に増える値として, この値をもとに,
`pc.printf("time: (%2d:%2d:%2d) ¥r",hour,min,sec);`
として, main関数の中で表示すること.
- また, 0.2秒毎のLEDの点滅は, 実行するが, 前述のリストの20行目はコメント行にして, printfが実行されないようにすること. (コメントにしないと, 20行目の表示が, main関数中の表示に割り込んでしまう)
- シリアルコンソール等で, 表示を確認し, 時間が正しく計測できていることを確認すること.



目次

- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- 8. LEDのスタティック点灯・ダイナミック点灯
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう

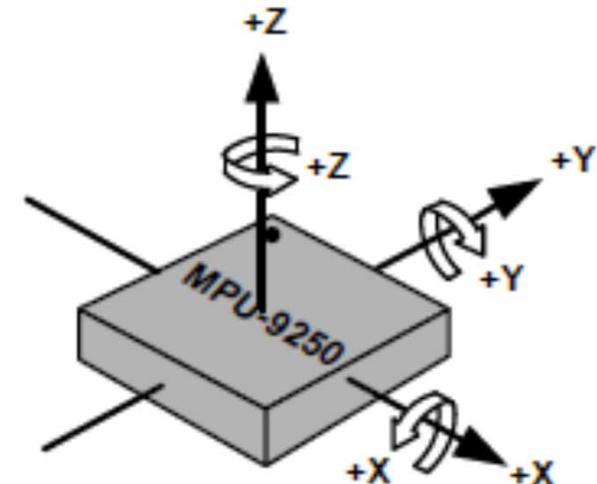


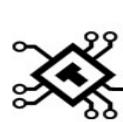
センサを読み込む

- バーサライタ基板には、9軸センサ MPU9250が搭載されている。
- 搭載されているジャイロの大まかな仕様は下表の通りである。

測定範囲	250/500/1000/2000 deg. / sec.
感度	131 / 65.5 / 32.8 / 16.4 LSB/ (dps)
データ幅	16ビット
データ出力	I2C/SPI

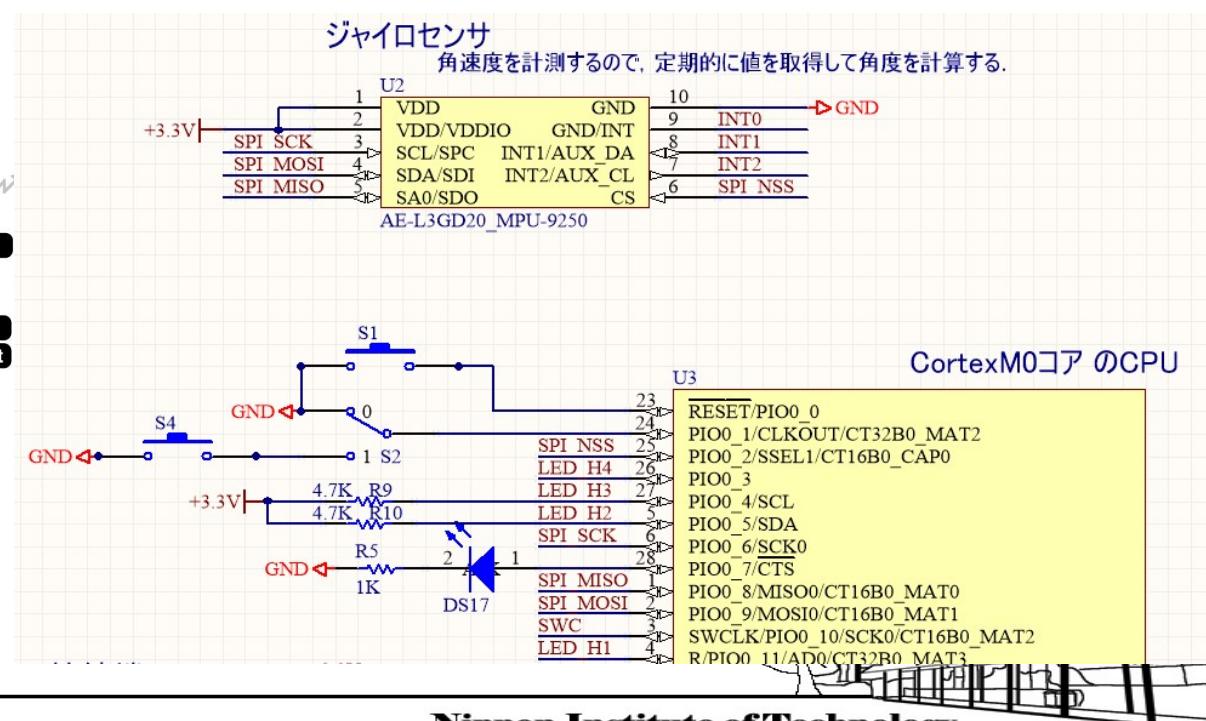
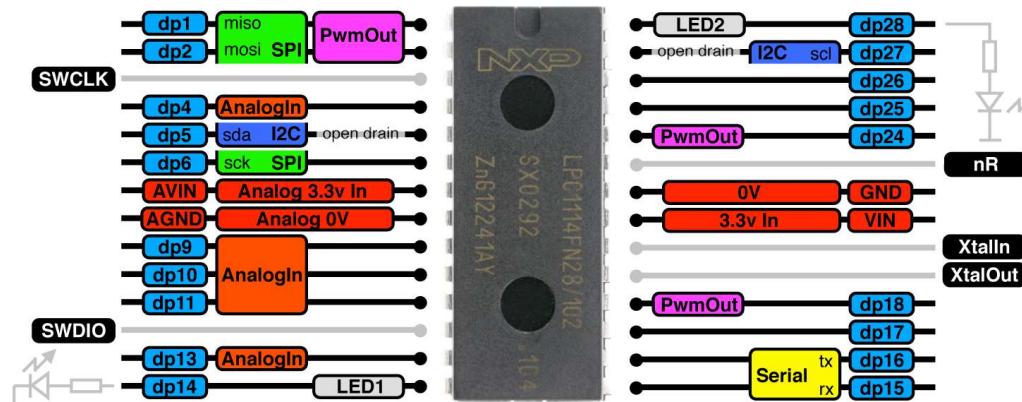
- チップ面に対して垂直なZ軸周りの回転速度を測定することで、バーサライタ基板の傾きを推測する。
- まず、センサとCPU間の通信を確認するため、SPI通信(単色版)またはI2C(フルカラー版)を実装する。
- SPI通信は、4本の信号を使う同期シリアル通信である。
- I2C通信は、2本の信号を使う同期シリアル通信である。

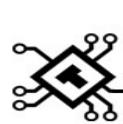




センサを読み込む(SPI)

- LPC1114FN28では、SPIは、1番ピン(MISO), 2番ピン(MOSI), 6番ピン(SCK)が割り当たっている。
- SPIの駆動には、もう一つチップセレクト信号(NSS)が必要である。
- mbed LPC1114FN28の設定では、チップセレクトは特に確保されていない。
- そこで、25番ピンのGPIOを使用して、通信を制御する。



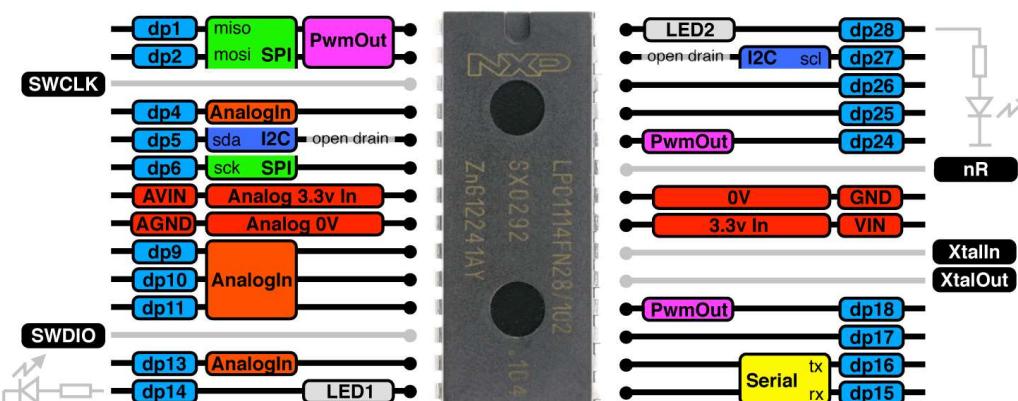


センサを読み込む(I2C)

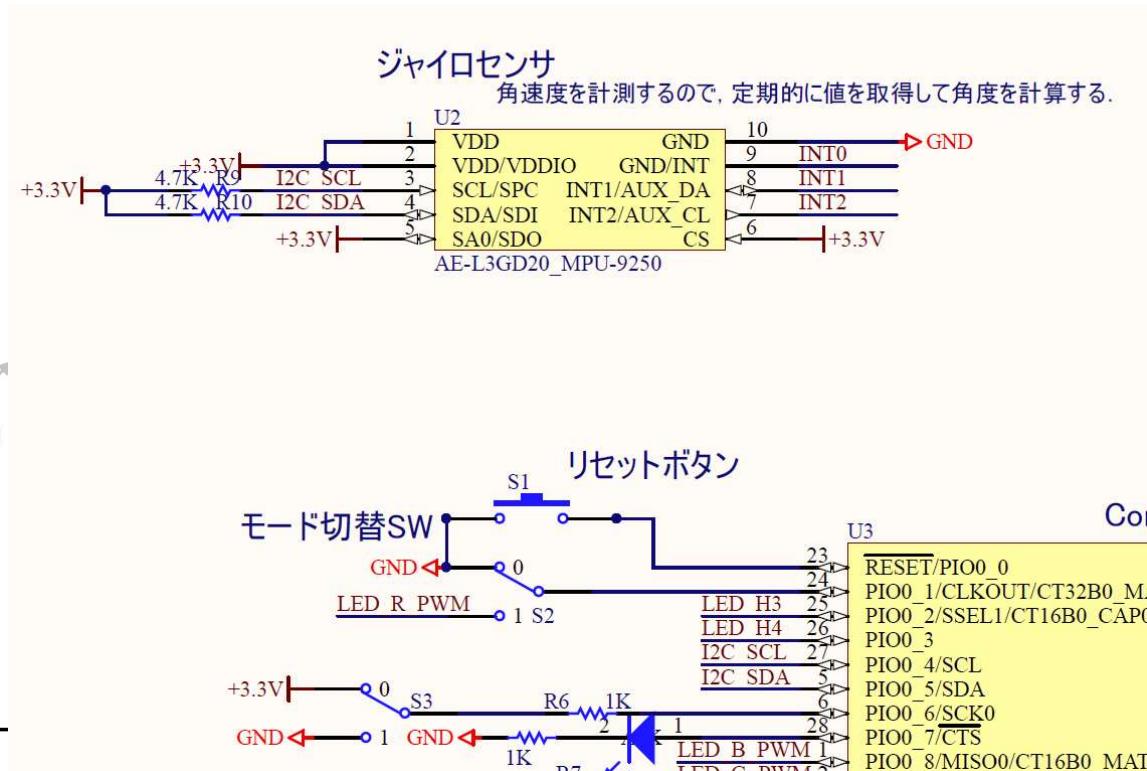
- LPC1114FN28では、I2Cは、27番ピン(SCL), 5番ピン(SDA)が割り当たっている。
- I2Cはオープンドレイン出力(電流を吸うだけで吐き出さない出力回路の形式)なので、出力をしていない状態を作るためにプルアップ抵抗(R9, R10)が必要である。
- I2Cでは通信対象毎にアドレスを使用してデータの送り先を指定する。
- 本回路では、ジャイロセンサのアドレスは

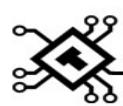
0x69

である



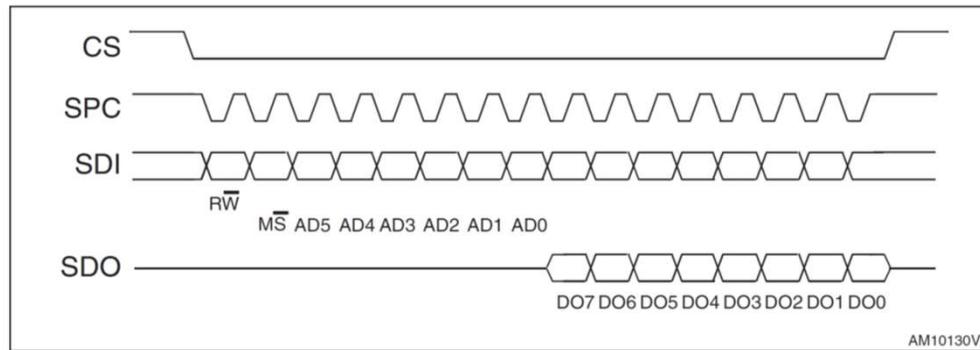
TAKITA Lab.





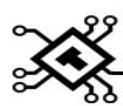
センサを読み込む(SPI)

- SPIでは、データの送受信は、クロック信号に同期して行なわれる。そのため、GPIOを利用して実装することが可能であるが、ここではmbedのSPIクラスを使用する。



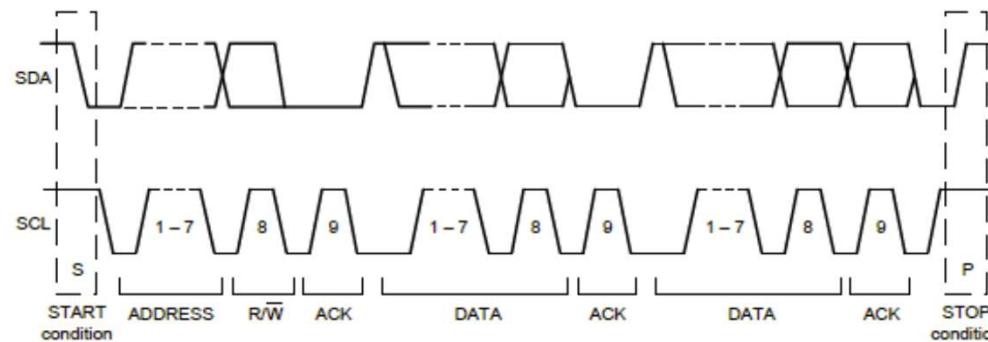
- 通信は、CSの信号を下げることで通信開始が通達され、そこから、クロックの立ち上がりで、データが取り込まれる。
- マイコンがマスターとなり、センサに指令を送るとそれに対応したデータが出力される。
- マスターからは、読み込み・書き込みの判別を行なうビット(R/W), アドレスの自動増加を指示するビット(M/S), それとアドレスビット(AD5～AD0)を最初に1バイト送信する。
- それに続くバイトは、読み込み・書き込みによって異なる。



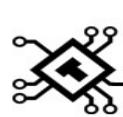


センサを読み込む(I2C)

- I2Cでは、データの送受信は、クロック信号に同期して行なわれる。ここではmbedのI2Cクラスを使用する。



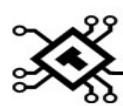
- 通信は、プルアップされた信号線に対して、SCLがHの時に、SDAを下げるSTART Conditionで通信開始が通達され、そこから、クロックの立ち上がりで、データが取り込まれる。
- マイコンがマスターとなり、センサに指令を送るとそれに対応したデータが出力される。
- マスターからは、読み込み・書き込みの判別を行なうビット(R/W)をつけたアドレスに対して、読み込み・書き込みの対象となるアドレスを指示する。
- それに続くバイトは、読み込み・書き込みによって異なる。



センサを読み込む

- 右表は、データシートに記載のレジスタマップである
- アドレスビットに右表のレジスタアドレスを指定するとそのレジスタ(メモリ)に書き込まれた情報を得ることが出来る。
- また、特定のレジスタに特定の値を書き込むことでセンサの設定を変更することが可能である。
- ここでは、バーサライタで使用するレジスタは、PWR_MGMT_1, GYRO_CONFIG, ACCEL_CONFIG, ACCEL_CONFIG2, GYRO_ZOUT_L, GYRO_ZOUT_Hである。
- まずは、テストのために、WHO_AM_レジスタを使用して、通信が出来ることを確認する。
- WHO_AM_レジスタは、書き込みの出来ないレジスタで、通信が正常に行なわれていれば、2進数で0b01110001, (16進数で0x71)が受信出来れば、通信は正常であることが分かる。

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F
00	0	SELF_TEST_X_GYRO	R/W	35	53	I2C_SLV4_DI	R	66	102	I2C_SLV3_DO	R/W
01	1	SELF_TEST_Y_GYRO	R/W	36	54	I2C_MST_STATUS	R	67	103	I2C_MST_DELAY_CTRL	R/W
02	2	SELF_TEST_Z_GYRO	R/W	37	55	INT_PIN_CFG	R/W	68	104	SIGNAL_PATH_RESET	R/W
0D	13	SELF_TEST_X_ACCEL	R/W	38	56	INT_ENABLE	R/W	69	105	MOT_DETECT_CTRL	R/W
0E	14	SELF_TEST_Y_ACCEL	R/W	3A	58	INT_STATUS	R	6A	106	USER_CTRL	R/W
0F	15	SELF_TEST_Z_ACCEL	R/W	3B	59	ACCEL_XOUT_H	R	6B	107	PWR_MGMT_1	R/W
13	19	XG_OFFSET_H	R/W	3C	60	ACCEL_XOUT_L	R	6C	108	PWR_MGMT_2	R/W
14	20	XG_OFFSET_L	R/W	3D	61	ACCEL_YOUT_H	R	72	114	FIFO_COUNTH	R/W
15	21	YG_OFFSET_H	R/W	3E	62	ACCEL_YOUT_L	R	73	115	FIFO_COUNTL	R/W
16	22	YG_OFFSET_L	R/W	3F	63	ACCEL_ZOUT_H	R	74	116	FIFO_R_W	R/W
17	23	ZG_OFFSET_H	R/W	40	64	ACCEL_ZOUT_L	R	75	117	WHO_AM_I	R
18	24	ZG_OFFSET_L	R/W	41	65	TEMP_OUT_H	R	77	119	XA_OFFSET_H	R/W
19	25	SMPLRT_DIV	R/W	42	66	TEMP_OUT_L	R	78	120	XA_OFFSET_L	R/W
1A	26	CONFIG	R/W	43	67	GYRO_XOUT_H	R	7A	122	YA_OFFSET_H	R/W
1B	27	GYRO_CONFIG	R/W	44	68	GYRO_XOUT_L	R	7B	123	YA_OFFSET_L	R/W
1C	28	ACCEL_CONFIG	R/W	45	69	GYRO_YOUT_H	R	7D	125	ZA_OFFSET_H	R/W
1D	29	ACCEL_CONFIG 2	R/W	46	70	GYRO_YOUT_L	R	7E	126	ZA_OFFSET_L	R/W
1E	30	LP_ACCEL_ODR	R/W	47	71	GYRO_ZOUT_H	R				
1F	31	WOM_THR	R/W	48	72	GYRO_ZOUT_L	R				
23	35	FIFO_EN	R/W	49	73	EXT_SENS_DATA_00	R				
24	36	I2C_MST_CTRL	R/W	4A	74	EXT_SENS_DATA_01	R				
25	37	I2C_SLV0_ADDR	R/W	4B	75	EXT_SENS_DATA_02	R				
26	38	I2C_SLV0_REG	R/W	4C	76	EXT_SENS_DATA_03	R				
27	39	I2C_SLV0_CTRL	R/W	4D	77	EXT_SENS_DATA_04	R				
28	40	I2C_SLV1_ADDR	R/W	4E	78	EXT_SENS_DATA_05	R				
29	41	I2C_SLV1_REG	R/W	4F	79	EXT_SENS_DATA_06	R				
2A	42	I2C_SLV1_CTRL	R/W	50	80	EXT_SENS_DATA_07	R				
2B	43	I2C_SLV2_ADDR	R/W	51	81	EXT_SENS_DATA_08	R				
2C	44	I2C_SLV2_REG	R/W	52	82	EXT_SENS_DATA_09	R				
2D	45	I2C_SLV2_CTRL	R/W	53	83	EXT_SENS_DATA_10	R				
2E	46	I2C_SLV3_ADDR	R/W	54	84	EXT_SENS_DATA_11	R				
2F	47	I2C_SLV3_REG	R/W	55	85	EXT_SENS_DATA_12	R				
30	48	I2C_SLV3_CTRL	R/W	56	86	EXT_SENS_DATA_13	R				
31	49	I2C_SLV4_ADDR	R/W	57	87	EXT_SENS_DATA_14	R				
32	50	I2C_SLV4_REG	R/W	58	88	EXT_SENS_DATA_15	R				
33	51	I2C_SLV4_DO	R/W	59	89	EXT_SENS_DATA_16	R				
34	52	I2C_SLV4_CTRL	R/W	5A	90	EXT_SENS_DATA_17	R				
				5B	91	EXT_SENS_DATA_18	R				
				5C	92	EXT_SENS_DATA_19	R				
				5D	93	EXT_SENS_DATA_20	R				
				5E	94	EXT_SENS_DATA_21	R				
				5F	95	EXT_SENS_DATA_22	R				
				60	96	EXT_SENS_DATA_23	R				
				63	99	I2C_SLV0_DO	R/W				
				64	100	I2C_SLV1_DO	R/W				
				65	101	I2C_SLV2_DO	R/W				



センサを読み込む

- ジャイロにアクセスするための情報を記録したMPU9250.hを作成する。

```
#ifndef __MPU9250_H__
#define __MPU9250_H__

#include "mbed.h"
#include "math.h"

// Original Information from RM-MPU-9250A-00
// https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU-9250-Register-Map.pdf

#define MPU9250_I2C_ADDR 0x69

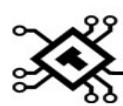
// Register Map for Gyroscope and Accelerometer
#define MPU9250_SELF_TEST_X_GYRO 0x00 // Gyroscope Self-Test Registers
#define MPU9250_SELF_TEST_Y_GYRO 0x01
#define MPU9250_SELF_TEST_Z_GYRO 0x02
#define MPU9250_SELF_TEST_X_ACCEL 0x0D // Accelerometer Self-Test Registers
#define MPU9250_SELF_TEST_Y_ACCEL 0x0E
#define MPU9250_SELF_TEST_Z_ACCEL 0x0F

#define MPU9250_XG_OFFSET_H 0x13 // Gyro Offset Registers
#define MPU9250_XG_OFFSET_L 0x14
#define MPU9250_YG_OFFSET_H 0x15
#define MPU9250_YG_OFFSET_L 0x16
#define MPU9250_ZG_OFFSET_H 0x17
#define MPU9250_ZG_OFFSET_L 0x18

#define MPU9250_SMPLRT_DIV 0x19 // Sample Rate Divider
#define MPU9250_CONFIG 0x1A // Configuration
#define MPU9250_GYRO_CONFIG 0x1B // Gyroscope Configuration
#define MPU9250_ACCEL_CONFIG 0x1C // Accelerometer Configuration
#define MPU9250_ACCEL_CONFIG2 0x1D // Accelerometer Configuration 2
#define MPU9250_LP_ACCEL_ODR 0x1E // Low Power Accelerometer ODR Control
#define MPU9250_WOM_THR 0x1F // Wake-on Motion Threshold
```

```
#define MPU9250_FIFO_EN 0x23 // FIFO Enable
#define MPU9250_I2C_MST_CTRL 0x24 // I2C Master Control
// I2C Slave 0 Control
#define MPU9250_I2C_SLV0_ADDR 0x25
#define MPU9250_I2C_SLV0_REG 0x26
#define MPU9250_I2C_SLV0_CTRL 0x27
// I2C Slave 1 Control
#define MPU9250_I2C_SLV1_ADDR 0x28
#define MPU9250_I2C_SLV1_REG 0x29
#define MPU9250_I2C_SLV1_CTRL 0x2A
// I2C Slave 2 Control
#define MPU9250_I2C_SLV2_ADDR 0x2B
#define MPU9250_I2C_SLV2_REG 0x2C
#define MPU9250_I2C_SLV2_CTRL 0x2D
// I2C Slave 3 Control
#define MPU9250_I2C_SLV3_ADDR 0x2E
#define MPU9250_I2C_SLV3_REG 0x2F
#define MPU9250_I2C_SLV3_CTRL 0x30
// I2C Slave 4 Control
#define MPU9250_I2C_SLV4_ADDR 0x31
#define MPU9250_I2C_SLV4_REG 0x32
#define MPU9250_I2C_SLV4_DO 0x33
#define MPU9250_I2C_SLV4_CTRL 0x34
#define MPU9250_I2C_SLV4_DI 0x35

#define MPU9250_I2C_MST_STATUS 0x36 // I2C Master Status
#define MPU9250_INT_PIN_CFG 0x37 // INT Pin/Bypass Enable Configuration
#define MPU9250_INT_ENABLE 0x38 // Interrupt Enable
#define MPU9250_INT_STATUS 0x3A // Interrupt Status
```



センサを読み込む

- 前ページの続き

```
// Accelerometer Measurements
#define MPU9250_ACCEL_XOUT_H 0x3B
#define MPU9250_ACCEL_XOUT_L 0x3C
#define MPU9250_ACCEL_YOUT_H 0x3D
#define MPU9250_ACCEL_YOUT_L 0x3E
#define MPU9250_ACCEL_ZOUT_H 0x3F
#define MPU9250_ACCEL_ZOUT_L 0x40
    // Temperature Measurement
#define MPU9250_TEMP_OUT_H 0x41
#define MPU9250_TEMP_OUT_L 0x42
    // Gyroscope Measurements
#define MPU9250_GYRO_XOUT_H 0x43
#define MPU9250_GYRO_XOUT_L 0x44
#define MPU9250_GYRO_YOUT_H 0x45
#define MPU9250_GYRO_YOUT_L 0x46
#define MPU9250_GYRO_ZOUT_H 0x47
#define MPU9250_GYRO_ZOUT_L 0x48
    // External Sensor Data
#define MPU9250_EXT_SENS_DATA_00 0x49
#define MPU9250_EXT_SENS_DATA_01 0x4A
#define MPU9250_EXT_SENS_DATA_02 0x4B
#define MPU9250_EXT_SENS_DATA_03 0x4C
#define MPU9250_EXT_SENS_DATA_04 0x4D
#define MPU9250_EXT_SENS_DATA_05 0x4E
#define MPU9250_EXT_SENS_DATA_06 0x4F
#define MPU9250_EXT_SENS_DATA_07 0x50
#define MPU9250_EXT_SENS_DATA_08 0x51
#define MPU9250_EXT_SENS_DATA_09 0x52
#define MPU9250_EXT_SENS_DATA_10 0x53
#define MPU9250_EXT_SENS_DATA_11 0x54
#define MPU9250_EXT_SENS_DATA_12 0x55
```

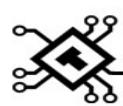
```
#define MPU9250_EXT_SENS_DATA_13 0x56
#define MPU9250_EXT_SENS_DATA_14 0x57
#define MPU9250_EXT_SENS_DATA_15 0x58
#define MPU9250_EXT_SENS_DATA_16 0x59
#define MPU9250_EXT_SENS_DATA_17 0x5A
#define MPU9250_EXT_SENS_DATA_18 0x5B
#define MPU9250_EXT_SENS_DATA_19 0x5C
#define MPU9250_EXT_SENS_DATA_20 0x5D
#define MPU9250_EXT_SENS_DATA_21 0x5E
#define MPU9250_EXT_SENS_DATA_22 0x5F
#define MPU9250_EXT_SENS_DATA_23 0x60

#define MPU9250_I2C_SLV0_DO 0x63 // I2C Slave 0 Data Out
#define MPU9250_I2C_SLV1_DO 0x64 // I2C Slave 1 Data Out
#define MPU9250_I2C_SLV2_DO 0x65 // I2C Slave 2 Data Out
#define MPU9250_I2C_SLV3_DO 0x66 // I2C Slave 3 Data Out
#define MPU9250_I2C_MST_DELAY_CTRL 0x67 // I2C Master Delay Control

#define MPU9250_SIGNAL_PATH_RESET 0x68 // Signal Path Reset

#define MPU9250_MOT_DETECT_CTRL 0x69 // Accelerometer Interrupt Control ACCEL_INTEL_CTRL
#define MPU9250_USER_CTRL 0x6A // User Control
#define MPU9250_PWR_MGMT_1 0x6B // Power Management 1 (Default Value 0x01)
#define MPU9250_PWR_MGMT_2 0x6C // Power Management 2
    // FIFO Count Registers
#define MPU9250_FIFO_COUNTH 0x72
#define MPU9250_FIFO_COUNTL 0x73
#define MPU9250_FIFO_R_W 0x74 // FIFO Read Write

#define MPU9250_WHO_AM_I 0x75 // WHO AM I (Default Value 0x71)
```



センサを読み込む

- 前ページの続き

```
// Accelerometer Offset Registers
#define I_AM_MPU9250 ((uint8_t)0x71)

#define MPU9250_XA_OFFSET_H 0x77
#define MPU9250_XA_OFFSET_L 0x78
#define MPU9250_YA_OFFSET_H 0x7A
#define MPU9250_YA_OFFSET_L 0x7B
#define MPU9250_ZA_OFFSET_H 0x7D
#define MPU9250_ZA_OFFSET_L 0x7E

//Magnetometer Registers
#define AK8963_I2C_ADDR 0x0C
#define AK8963_WIA 0x00 // Device ID (Default value 0x48)
#define AK8963_INFO 0x01 // Information
#define AK8963_ST1 0x02 // Status 1
    // Measurement data
#define AK8963_XOUT_L 0x03
#define AK8963_XOUT_H 0x04
#define AK8963_YOUT_L 0x05
#define AK8963_YOUT_H 0x06
#define AK8963_ZOUT_L 0x07
#define AK8963_ZOUT_H 0x08

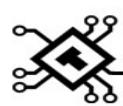
#define AK8963_ST2 0x09 // Status 2
#define AK8963_CNTL1 0x0A // Control1
#define AK8963_CNTL2 0x0B // Control2
#define AK8963_RSV 0x0C // Reserved
#define AK8963_ASTC 0x0C // Self-test
#define AK8963_I2CDIS 0x0F // I2C Disable
#define AK8963_ASAX 0x10 // X-axis sensitivity adjustment value
#define AK8963_ASAY 0x11 // Y-axis sensitivity adjustment value
#define AK8963_ASAZ 0x12 // Z-axis sensitivity adjustment value
```

```
#define MPU9250_H_RESET 0x80
#define MPU9250_SLEEP 0x40
#define MPU9250_CYCLE 0x20
#define MPU9250_GYRO_STADBY 0x10
#define MPU9250_PD_PTAT 0x08
#define MPU9250_CLKSEL_STOP 0x07
#define MPU9250_CLKSEL_AUTO 0x03
#define MPU9250_CLKSEL_I20 0x00
#define MPU9250_BYPASS_EN 0x02

#define MPI9250_GFS_250 (0x00<<3)
#define MPI9250_GFS_500 (0x01<<3)
#define MPI9250_GFS_1000 (0x02<<3)
#define MPI9250_GFS_2000 (0x03<<3)

#define MPI9250_AFS_2G (0x00<<3)
#define MPI9250_AFS_4G (0x01<<3)
#define MPI9250_AFS_8G (0x02<<3)
#define MPI9250_AFS_16G (0x03<<3)

#define MPI9250_GBW_250 0x00
#define MPI9250_GBW_184 0x01
#define MPI9250_GBW_92 0x02
#define MPI9250_GBW_41 0x03
#define MPI9250_GBW_20 0x04
#define MPI9250_GBW_10 0x05
#define MPI9250_GBW_5 0x06
```



センサを読み込む

- 前ページの続き

```
#define MPI9250_ABW_460 0x00
#define MPI9250_ABW_184 0x01
#define MPI9250_ABW_92 0x02
#define MPI9250_ABW_41 0x03
#define MPI9250_ABW_20 0x04
#define MPI9250_ABW_10 0x05
#define MPI9250_ABW_5 0x06
//#define MPI9250_ABW_460 0x07

#define MPU9250_WRITE 0x00
#define MPU9250_READ 0x80

// Set initial input parameters
enum MPU9250_ACCEL_FS {
    ACCEL_FS_SEL_2G = 0,
    ACCEL_FS_SEL_4G,
    ACCEL_FS_SEL_8G,
    ACCEL_FS_SEL_16G
};

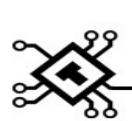
enum MPU9250_GYRO_FS {
    GYRO_FS_SEL_250DPS = 0,
    GYRO_FS_SEL_500DPS,
    GYRO_FS_SEL_1000DPS,
    GYRO_FS_SEL_2000DPS
};
```

```
enum MPU9250_MAG_BIT {
    MAG_BIT_14BITS = 0, // 0.6uT/LSB
    MAG_BIT_16BITS // 0.15uT/LSB
};

#define MPU9250_DLDPF_CFG_256HZ_NOLPF2 0x00
#define MPU9250_DLDPF_CFG_184HZ 0x01 // Low-pass filter Gyro 184Hz, Temp
#define MPU9250_DLDPF_CFG_98HZ 0x02
#define MPU9250_DLDPF_CFG_42HZ 0x03
#define MPU9250_DLDPF_CFG_20HZ 0x04
#define MPU9250_DLDPF_CFG_10HZ 0x05
#define MPU9250_DLDPF_CFG_5HZ 0x06
#define MPU9250_DLDPF_CFG_2100HZ_NOLPF 0x07

#define AK8963_MODE_PWRDOWN 0x00 // Power-down mode
#define AK8963_MODE_SNGLMES 0x01 // Single measurement mode
#define AK8963_MODE_CONMES1 0x02 // Continuous measurement mode 1
#define AK8963_MODE_CONMES2 0x06 // Continuous measurement mode 2
#define AK8963_MODE_EXTRIG 0x04 // External trigger measurement mode
#define AK8963_MODE_SELFTST 0x08 // Self-test mode
#define AK8963_MODE_FUSEROM 0x0F // Fuse ROM access mode

#endif
```

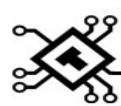


センサを読み込む

- 2行目で、前述のヘッダーファイルを読み込む。
- 4行目が、通信方式の選択。
- 9～11行目がSPI・NSS(チップセレクト)の設定である。
- SPIはMOSI,MISO,SCLKの順に引数に該当するピンを与える。
- 12～13行目がI2Cの設定である。
- I2Cは、SDA, SCLの順に引数に該当するピンを与える。
- 24行目がSPIの通信フォーマットを決めている。最初の引数が通信の区切りとなるビット数、2番目がSPIのモード番号(後述)である。
- 25行目でSPIの通信速度を設定している。.
- 26行目でチップセレクトをHighにしている。
- 28行目でI2Cの通信速度を400kbpsに設定している。
- 34～37行目・39～42行目が実際の通信である。
- 読み取りたいアドレスを送り、次のバイトを返信として受信する。
- 受信した結果が右図である。

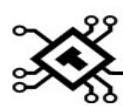
LPC1114 Light Stick Demo
WHOAMI register = 0x71
WHOAMI register = 0x 71

```
1. #include "mbed.h"
2. #include "MPU9250.h"
3.
4. #define USE_SPI // USE_I2C
5.
6. DigitalOut myled1(LED1);
7. DigitalOut myled2(LED2);
8. Serial pc(dp16, dp15); // tx, rx
9. #ifdef USE_SPI
10. SPI gyro(dp2, dp1, dp6);
11. DigitalOut sscs(dp25);
12. #else
13. I2C gyro(dp5, dp27);
14. #endif
15.
16. int main() {
17.     int ret;
18.     char tmp;
19.     pc.baud(115200);
20.     pc.printf("LPC1114 Light Stick Demo\r\n");
21.
22.     myled1 = myled2 = 0;
23. #ifdef USE_SPI
24.     gyro.format(8,3);
25.     gyro.frequency(5000000);
26.     sscs = 1;
27. #else
28.     gyro.frequency(400000);
29. #endif
30.     myled1 = 1;
31.
32.     while(1) {
33. #ifdef USE_SPI
34.     sscs = 0 ;
35.     gyro.write(MPU9250_READ | MPU9250_WHO_AM_I);
36.     ret = gyro.read(0x00);
37.     sscs = 1;
38. #else
39.     tmp = MPU9250_WHO_AM_I ;
40.     gyro.write((int)MPU9250_I2C_ADDR<<1,(char*)&tmp, 1, 1); // no stop
41.     gyro.read((int)MPU9250_I2C_ADDR<<1,(char*)&tmp, 1, 0);
42.     ret = tmp ;
43. #endif
44.     myled1 = myled1^1;
45.     myled2 = myled1^1;
46.     pc.printf("WHOAMI register = 0x%X\r\n", ret);
47.     wait(0.2);
48. }
49. }
```



センサを読み込む(初期化の概要)

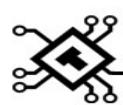
- ・センサから角速度の情報を読み込むためには, PWR_MGMT_1, CONFIG, GYRO_CONFIG, ACCEL_CONFIG, ACCEL_CONFIG2などを適切に設定しなければならない.
- ・また, GYRO_ZOUT_L, GYRO_ZOUT_Hからは, 下位バイト・上位バイトが取得できるので, この二つを組み合わせて16ビット信号にしないと測定値が得られない.
- ・PWR_MGMT_1には, H_RESETをonにして, リセット後に, 0x00を設定する.
- ・CONFIGは, サンプリング周波数8kHz, カットオフ周波数を250Hzに設定する.
- ・SMPLRT_DIVには0を設定する.
- ・GYRO_CONFIGのFCHOICE_Bは0x00に, 測定レンジを2000dpsに設定する.
- ・以上の設定で連続してZ軸周りのデータが計測される.



センサを読み込む

- 22～44行目が SPI接続時のセンサと通信を行うための基本関数である。
- SPI通信は、クロックに同期してデータが送受信される通信方式である。
- 23～32行目の関数では、CSを1→0に変化させることでセンサに通信開始を知らせ、センサに操作対象のアドレスを送り、クロックに同期して送られてくる返信を取り込んでいる。

```
1. #include "mbed.h"
2. #include "MPU9250.h"
3. #define D2D_MPU9250 (0.06097561)
4. #define USE_SPI // USE_I2C
5.
6. #ifdef USE_SPI
7.     SPI gyro(dp2, dp1, dp6);
8.     DigitalOut scs(dp25);
9. #else
10.    I2C gyro(dp5, dp27);
11. #endif
12.
13. DigitalOut myled1(LED1);
14. DigitalOut myled2(LED2);
15. Serial pc(dp16, dp15); // tx, rx
16.
17. uint8_t gyro_i2c_addr = MPU9250_I2C_ADDR;
18. uint8_t gyro_l_addr = MPU9250_GYRO_ZOUT_L, gyro_h_addr = MPU9250_GYRO_ZOUT_H;
19. uint8_t gyro_read_flg = MPU9250_READ, gyro_write_flg = MPU9250_WRITE;
20. float dig2dps = D2D_MPU9250;
21.
22. #ifdef USE_SPI
23.     uint8_t gyro_read_write(uint8_t addr,uint8_t dat)
24. {
25.     uint8_t ret;
26.     scs = 1;
27.     scs = 0 ;
28.     gyro.write(addr);
29.     ret = gyro.write(dat);
30.     scs = 1;
31.     return ret;
32. }
33.
34. void gyro_write(uint8_t addr,uint8_t dat)
35. {
36.     gyro_read_write(addr|gyro_write_flg,dat);
37. }
38.
39. uint8_t gyro_read(uint8_t addr)
40. {
41.     uint8_t ret ;
```



センサを読み込む

- 34～37行目は、前述の関数を書き込みに特化した関数である。
- 39～44行目は、前述の関数を読み込みに特化した関数である。
- 46～60行目が、I2C接続時の読み書きのための関数である。
- I2Cもクロックに同期して送受を行うが、CS信号を使用しない2線式の通信方式である。
- CSの代わりに送受する相手先のデバイスのID(gyro_i2c_addr)を最初に送信する。
- 63行目からが、センサの初期化である。

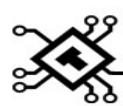
```
33.
34. void gyro_write(uint8_t addr,uint8_t dat)
35. {
36.   gyro_read_write(addr|gyro_write_flg,dat);
37. }
38.
39. uint8_t gyro_read(uint8_t addr)
40. {
41.   uint8_t ret ;
42.   ret = gyro_read_write(addr|gyro_read_flg,0x00);
43.   return ret;
44. }
45. #else
46. void gyro_write(uint8_t addr,uint8_t dat)
47. {
48.   uint8_t sdat[2];
49.   sdat[0] = addr ;
50.   sdat[1] = dat ;
51.   gyro.write((int)gyro_i2c_addr<<1,(char*)sdat, 2, 0); // stop
52. }
53.
54. uint8_t gyro_read(uint8_t addr)
55. {
56.   uint8_t ret ;
57.   gyro.write((int)gyro_i2c_addr<<1,(char*)&addr, 1, 1); // no stop
58.   gyro.read((int)gyro_i2c_addr<<1,(char*)&ret, 1, 0);
59.   return ret;
60. }
61. #endif
62.
63. void gyro_init(void)
64. {
65.   int ret ;
66.   uint8_t mpu_adr = 0;
67.   uint8_t mpu_reg = 0;
68.
69.   myled1 = myled2 = 0;
70.
71. #ifdef USE_SPI
72.   gyro.format(8,3);
73.   gyro.frequency(5000000);
```



センサを読み込む

- 71～74行目, 76行目がそれぞれの通信の初期設定を行う。
- SPIでは、速度・クロックとデータの組み合わせ・1データ分のクロック数などを設定し、CSを1にして火通信状態にする。
- I2Cでは、通信速度だけを設定する。
- 79～87行目では、接続されたセンサがMPU9250か確認している。
- 確認後、89行目から初期化の処理が開始される。
- 90～102行目でPWR_MGMT_1をリセットを設定し、センサを初期化する。

```
62.  
63. void gyro_init(void)  
64. {  
65.     int ret ;  
66.     uint8_t mpu_adr = 0;  
67.     uint8_t mpu_reg = 0;  
68.  
69.     myled1 = myled2 = 0;  
70.  
71. #ifdef USE_SPI  
72.     gyro.format(8,3);  
73.     gyro.frequency(5000000);  
74.     scs = 1;  
75. #else  
76.     gyro.frequency(400000);  
77. #endif  
78.  
79.     ret = gyro_read(MPU9250_WHO_AM_I);  
80.     pc.printf("WHOAMI register = 0%X\n\r", ret);  
81.     if(ret == I_AM_MPU9250) {  
82.         pc.printf("MPU9250 is found\n\r");  
83.         myled1 = 1;  
84.     }else{  
85.         pc.printf("no MPU9250 is found\n\r");  
86.         return ;  
87.     }  
88.  
89.     do{  
90.         mpu_adr = MPU9250_PWR_MGMT_1 ;  
91.         mpu_reg = MPU9250_H_RESET ;  
92.         gyro_write(mpu_adr,mpu_reg);  
93.         wait_ms(10);  
94.  
95.         mpu_adr = MPU9250_PWR_MGMT_1 ;  
96.         mpu_reg = 0x00 ;  
97.         gyro_write(mpu_adr,mpu_reg);  
98.  
99.         mpu_adr = MPU9250_PWR_MGMT_1 ;  
100.        ret=gyro_read(mpu_adr);  
101.    }while(ret != mpu_reg);  
102.
```



センサを読み込む

- 104～115行目で、測定条件を設定し、ジャイロを起動する。
- ジャイロの測定レンジは、 $\pm 2000\text{dps}$ である。
- 121行目からがmain文である。

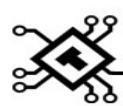
```
86.     return;
87. }
88.
89. do{
90.     mpu_adr = MPU9250_PWR_MGMT_1 ;
91.     mpu_reg = MPU9250_H_RESET ;
92.     gyro_write(mpu_adr,mpu_reg);
93.     wait_ms(10);
94.
95.     mpu_adr = MPU9250_PWR_MGMT_1 ;
96.     mpu_reg = 0x00 ;
97.     gyro_write(mpu_adr,mpu_reg);
98.
99.     mpu_adr = MPU9250_PWR_MGMT_1 ;
100.    ret=gyro_read(mpu_adr);
101.
102. }while(ret != mpu_reg);
103.
104. gyro_write(MPU9250_CONFIG,MPI9250_GBW_250);
105. gyro_write(MPU9250_SMPLRT_DIV, 0x00);
106.
107. do{
108.     mpu_adr = MPU9250_GYRO_CONFIG ;
109.     ret = gyro_read(mpu_adr);
110.     ret = ret & ~0x03; // Clear Fchoice bits [1:0]
111.     ret = ret & ~0x18; // Clear FS bits [4:3]
112.     mpu_reg = ret | MPI9250_GFS_2000; // Set full scale range for the gyro
113.     gyro_write(mpu_adr,mpu_reg);
114.     ret = gyro_read(mpu_adr);
115. }while(ret != mpu_reg);
116.
117. myled2 = 1;
118. pc.printf("I did it \n\r");
119.}
120.
121.int main(void)
122.{
123.     int32_t z_data;
124.     uint16_t tz_data;
125.     uint8_t dmy;
126.
127.     pc.baud(115200);
```



センサを読み込む

- 121行目からのmain関数のwhileループの中で、Z軸周りの角速度を計測している。
- 16ビットで計測される角速度を下位バイト(136～137行)・上位バイト(138～139行)で取得している。
- それぞれのバイトは、137行目、139行目で仮変数に統合され、140行目で符号付きの変数に代入されている。
- 142行目のprintfでは、感度の係数であるdig2dps(値としては、D2D_MP9250で定義したもののが代入されている)をかけて、計測結果を1秒あたりの角度変化に変換している。

```
115. }while(ret != mpu_reg);
116.
117. myled2 = 1;
118. pc.printf("I did it ¥n¥r");
119.}
120.
121.int main(void)
122.{
123. int32_t z_data;
124. uint16_t tz_data;
125. uint8_t dmy;
126.
127. pc.baud(115200);
128. pc.printf("LPC1114 Light Stick Demo¥r¥n");
129.
130. gyro_init();
131.
132. myled1 = 1;
133. myled2 = 0;
134.
135. while(1){
136.     dmy=gyro_read(gyro_l_addr);
137.     tz_data = (0x00ff&dmy);
138.     dmy=gyro_read(gyro_h_addr);
139.     tz_data |= (0xff00&(dmy<<8));
140.     z_data = (int16_t)(tz_data);
141.
142.     pc.printf("Z_data = %lf ¥n¥r", (double)z_data*dig2dps);
143.
144.     myled1 = myled1^1;
145.     myled2 = myled1^1;
146.//     wait(0.2);
147. }
148.}
```

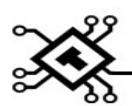


センサを読み込む(課題)

- MPU9250には、-40～85 °Cが測定可能な温度センサが搭載されている。
- TEMP_OUT_H, TEMP_OUT_Lを取得し、16ビットの符号付きデータとして変換後に以下の式に代入することで温度を取得可能である。
- 温度を取得するプログラムを作成し、MPU9250の上に指をあて温度が測定できることを確認せよ。

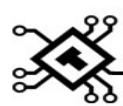
発展課題

- 本稿では、ジャイロの使用を想定している。
- 加速度センサについてもACCEL_CONFIG, ACCEL_CONFIG 2を設定すること使用が可能である。
- 加速度センサを設定して、ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H, ACCEL_ZOUT_Lを読むプログラムを作成してみること。



目次

- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- 8. LEDのスタティック点灯・ダイナミック点灯
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう



センサとタイマを使った角度計算

- ・ジャイロセンサから取得できるのは、角速度である。
- ・一方、処理に有用な情報は角度であることがおおい。
- ・そこで、速度から角度に変換する。→ 積分 $\int \omega(t)dt$ を求める。
- ・コンピュータでは連続した時間を処理できない。
- ・A/D変換には決まった時間で行なわれる。
- ・積分を連続した関数を解析的に解く → 離散的に取得される値を数値的に解く。
- ・積分 → 離散的に観測された値 × 取得時間の間隔の累積で考える。

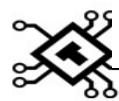
$$\int \omega(t)dt \rightarrow \sum \omega_i \Delta t$$

- ・取得された観測値に観測する時間間隔をかけて、足すことで、角度を求める。



センサとタイマを使った角度計算

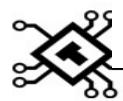
- Tickerを1msに設定し, 1ms毎にデータを取得する.
- 取得したデータに取得周期である1msをかけて足しあわせて, 角度データを作成する.
- 計算した角度をシリアルでPCに転送して表示させる.
- プログラムを次のスライドに示す.
- 16行目で, Tickerを宣言し, 12行目で角度を格納する変数を宣言している.
- 25~46行目がSPIの通信を行なう関数である.
- 48~62行目がI2Cの通信を行なう関数である.
- 65~120行目がジャイロの初期化関数である.
- 122~135行目が1ms毎にデータを取得し, 角度に変換している処理である.
- 実行してみると, 時間とともに, 誤差が蓄積されていく.
- 誤差の原因是, 主に, バイアス・温度ドリフト・ノイズである. そのうち積分に強く影響を与えているのは, バイアスである.



センサとタイマを使った角度計算

```
1. #include "mbed.h"
2. #include "MPU9250.h"
3. #define D2D_MPU9250 (0.06097561)
4. #define USE_SPI // USE_I2C
5.
6. #ifdef USE_SPI
7.     SPI gyro(dp2, dp1, dp6);
8.     DigitalOut scs(dp25);
9. #else
10.    I2C gyro(dp5, dp27);
11. #endif
12.
13. DigitalOut myled1(LED1);
14. DigitalOut myled2(LED2);
15. Serial pc(dp16, dp15); // tx, rx
16. Ticker gyro_handler;
17.
18. uint8_t gyro_i2c_addr = MPU9250_I2C_ADDR;
19. uint8_t gyro_l_addr = MPU9250_GYRO_ZOUT_L, gyro_h_addr = MPU9250_GYRO_ZOUT_H;
20. uint8_t gyro_read_flg = MPU9250_READ, gyro_write_flg = MPU9250_WRITE;
21. float dig2dps = D2D_MPU9250;
22. float angle = 0.0;
23.
24. #ifdef USE_SPI
25.     uint8_t gyro_read_write(uint8_t addr,uint8_t dat)
26. {
27.     uint8_t ret;
28.     scs = 1;
29.     scs = 0 ;
30.     gyro.write(addr);
31.     ret = gyro.write(dat);
32.     scs = 1;
33.     return ret;
34. }
35.
```

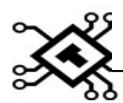
```
35.
36.     void gyro_write(uint8_t addr,uint8_t dat)
37. {
38.     gyro_read_write(addr|gyro_write_flg,dat);
39. }
40.
41.     uint8_t gyro_read(uint8_t addr)
42. {
43.     uint8_t ret ;
44.     ret = gyro_read_write(addr|gyro_read_flg,0x00);
45.     return ret;
46. }
47. #else
48.     void gyro_write(uint8_t addr,uint8_t dat)
49. {
50.     uint8_t sdat[2];
51.     sdat[0] = addr ;
52.     sdat[1] = dat ;
53.     gyro.write((int)gyro_i2c_addr<<1,(char*)sdat, 2, 0); // stop
54. }
55.
56.     uint8_t gyro_read(uint8_t addr)
57. {
58.     uint8_t ret ;
59.     gyro.write((int)gyro_i2c_addr<<1,(char*)&addr, 1, 1); // no stop
60.     gyro.read((int)gyro_i2c_addr<<1,(char*)&ret, 1, 0);
61.     return ret;
62. }
63. #endif
64.
65.     void gyro_init(void)
66. {
67.     int ret ;
68.     uint8_t mpu_adr = 0;
69.     uint8_t mpu_reg = 0;
```



センサとタイマを使った角度計算(つづき)

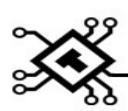
```
64.  
65. void gyro_init(void)  
66. {  
67.     int ret ;  
68.     uint8_t mpu_adr = 0;  
69.     uint8_t mpu_reg = 0;  
70.  
71.     myled1 = myled2 = 0;  
72.  
73. #ifdef USE_SPI  
74.     gyro.format(8,3);  
75.     gyro.frequency(5000000);  
76.     scs = 1;  
77. #else  
78.     gyro.frequency(400000);  
79. #endif  
80.  
81.     ret = gyro_read(MPU9250_WHO_AM_I);  
82.     pc.printf("WHOAMI register = 0x%X \n\r", ret);  
83.     if(ret == I_AM_MPU9250) {  
84.         pc.printf("MPU9250 is found \n\r");  
85.         myled1 = 1;  
86.     }else{  
87.         pc.printf("no MPU9250 is found \n\r");  
88.         return ;  
89.     }  
90.  
91.     do{  
92.         mpu_adr = MPU9250_PWR_MGMT_1 ;  
93.         mpu_reg = MPU9250_H_RESET ;  
94.         gyro_write(mpu_adr,mpu_reg);  
95.         wait_ms(10);  
96.  
97.         mpu_adr = MPU9250_PWR_MGMT_1 ;  
98.         mpu_reg = 0x00 ;
```

```
96.  
97.         mpu_adr = MPU9250_PWR_MGMT_1 ;  
98.         mpu_reg = 0x00 ;  
99.         gyro_write(mpu_adr,mpu_reg);  
100.  
101.        mpu_adr = MPU9250_PWR_MGMT_1 ;  
102.        ret=gyro_read(mpu_adr);  
103.    }while(ret != mpu_reg);  
104.  
105.    gyro_write(MPU9250_CONFIG,MPI9250_GBW_250);  
106.    gyro_write(MPU9250_SMPLRT_DIV, 0x00);  
107.  
108.    do{  
109.        mpu_adr = MPU9250_GYRO_CONFIG ;  
110.        ret = gyro_read(mpu_adr);  
111.        ret = ret & ~0x03; // Clear Fchoice bits [1:0]  
112.        ret = ret & ~0x18; // Clear FS bits [4:3]  
113.        mpu_reg = ret | MPI9250_GFS_2000; // Set full scale range for the gyro  
114.        gyro_write(mpu_adr,mpu_reg);  
115.        ret = gyro_read(mpu_adr);  
116.    }while(ret != mpu_reg);  
117.  
118.    myled2 = 1;  
119.    pc.printf("I did it \n\r");  
120.}  
121.  
122. void gyro_int(void)  
123.{  
124.     uint16_t tz_data;  
125.     int32_t z_data;  
126.     uint8_t dmy;  
127.  
128.     dmy=gyro_read(gyro_I_addr);  
129.     tz_data = (0x00ff&dmy);  
130.     dmy=gyro read(gyro_h_addr);
```

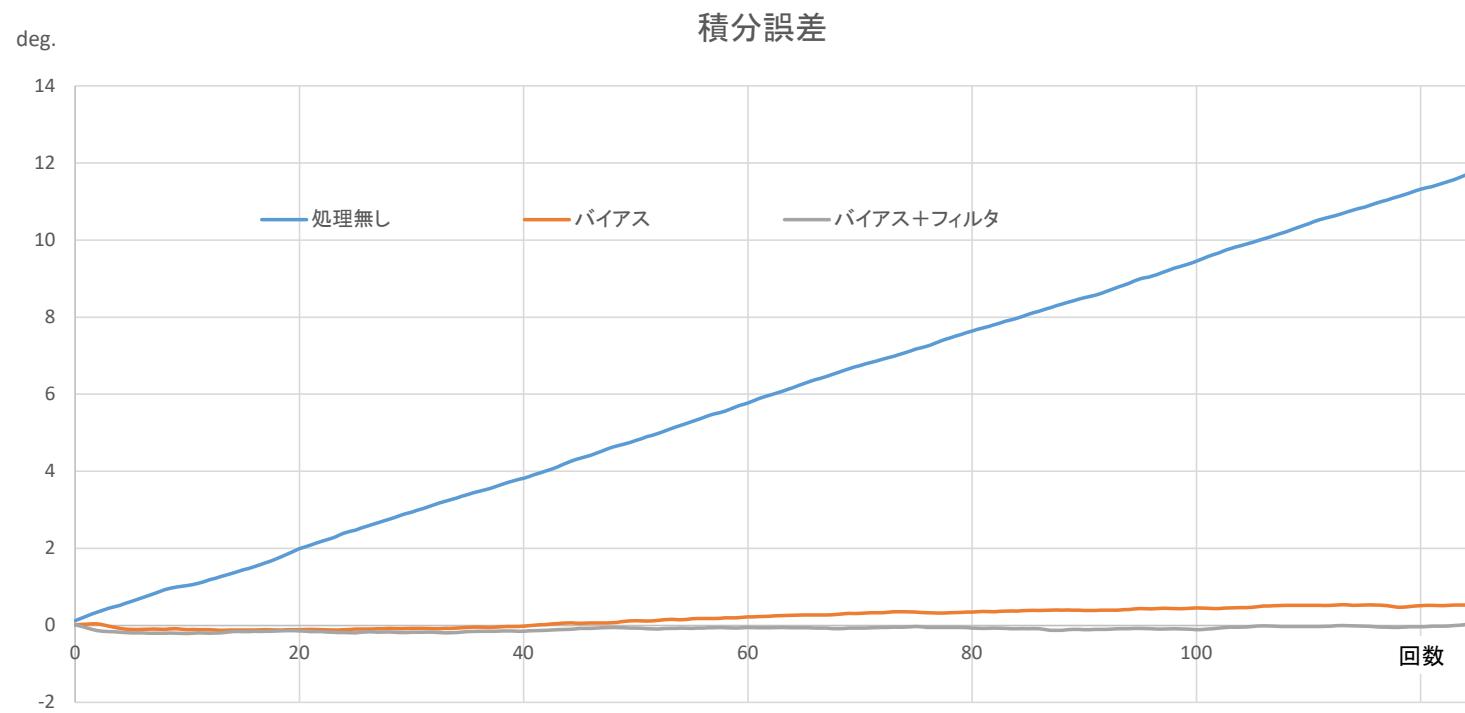


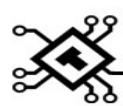
センサとタイマを使った角度計算(つづき)

```
127.
128. dmy=gyro_read(gyro_l_addr);
129. tz_data = (0x00ff&dmy);
130. dmy=gyro_read(gyro_h_addr);
131. tz_data |= (0xff00&(dmy<<8));
132. z_data = (int16_t)tz_data;
133.
134. angle += ((float)z_data*dig2dps)*0.001;
135.}
136.
137.int main(void)
138.{  
139. pc.baud(115200);
140. pc.printf("LPC1114 Light Stick Demo\r\n");
141.
142. gyro_init();
143. gyro_handler.attach_us(&gyro_int,1000);
144.
145. myled1 = 1;
146. myled2 = 0;
147.
148. while(1){
149.   pc.printf("angle = %f \r\n",angle);
150.   myled1 = myled1^1;
151.   myled2 = myled1^1;
152. }
153.}
```



センサとタイマを使った角度計算



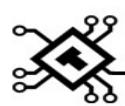


センサとタイマを使った角度計算(3)

```
1. #include "mbed.h"
2. #include "L3gd20.h"
3.
4. #define DIG2DPS (0.070)
5. #define RET_RANGE (-429) // -30 dps
6.
7. DigitalOut myled1(LED1);
8. DigitalOut myled2(LED2);
9.
10. Serial pc(dp16, dp15); // tx, rx
11.
12. DigitalOut scs(dp25);
13. SPI gyro(dp2, dp1, dp6);
14. Ticker gyro_handler;
15. float angle;
16. float z_bias;
17.
18. uint8_t GYRO_READ_WRITE(uint8_t WRAddr, uint8_t sBuffer)
19. {
20. --- 前述と同じ ---
21. }
22.
23. void gyro_init(void)
24. {
25. --- 前述と同じ ---
26. }
27.
28. void gyro_bias(void)
29. {
30. --- 前述と同じ ---
31. }
32.
33. void gyro_int(void)
34. {
35. static uint16_t tz_data[4]={0,0,0,0};
36. int32_t z_data;
```

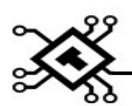
```
32. void gyro_int(void)
33. {
34. static uint16_t tz_data[4]={0,0,0,0};
35. int32_t z_data;
36. uint8_t dmy;
37. int16_t i;
38.
39. for(i=2;i>=0;i--){
40. tz_data[i+1] = tz_data[i] ;
41. }
42.
43. dmy=GYRO_READ_WRITE(L3GD20_OUT_Z_L_ADDR|L3GD20_READ,0x00);
44. tz_data[0] = (0x00ff&dmy);
45. dmy=GYRO_READ_WRITE(L3GD20_OUT_Z_H_ADDR|L3GD20_READ,0x00);
46. tz_data[0] |= (0xff00&(dmy<<8));
47.
48. z_data = (int16_t)tz_data[0];
49. for(i=1;i<4;i++){
50. z_data += (int16_t)tz_data[i];
51. }
52. z_data = z_data / 4;
53.
54. angle += ((float)z_data*DIG2DPS-z_bias)*0.001;
55.
56. if((int16_t)tz_data[0] < RET_RANGE)
57. if((int16_t)tz_data[1] < RET_RANGE)
58. if((int16_t)tz_data[2] < RET_RANGE)
59. if((int16_t)tz_data[3] < RET_RANGE)
60. angle = 0.0;
61. }
62.
63. int main() {
64. --- 前述と同じ ---
65. }
```





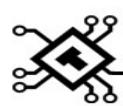
センサとタイマを使った角度計算(3)

- ・角度を求めるには、加速度計を使用する方法もある。
- ・それぞれ一長一短あるが、どのような組合せ方があるか調べること。



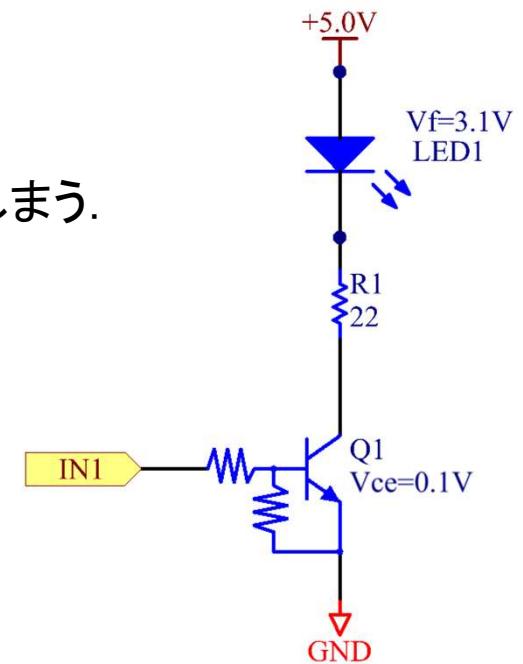
目次

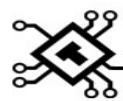
- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- **8. LEDのスタティック点灯・ダイナミック点灯**
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう



LEDのスタティック点灯・ダイナミック点灯

- LEDは低消費電力と言われているが、駆動回路(駆動方法)次第で消費電力は大きく異なる。
- 下図のようなLEDが並んだ回路があったとする。なお、トランジスタでの電圧降下は0.1Vである。
- 仮にLEDの順方向電圧 $V_f=3.0V$ とすると、抵抗での電圧降下は、1.9Vである。
- 一つの回路に流れる電流は、 $I=86mA$ である。
- 消費電力としては、 $86mA \times 5.0V \doteq 0.43W$ となる。
- LEDが16個あるとすると、6.88Wとなる。
- (700gの物体が毎秒1mの速度で移動するのと等しい仕事率)
- モバイルバッテリが2000mAhとすると、大体1時間ぐらいで空っぽになってしまう。
- LEDを見るのは人間である。
→ 点灯してると人の目が認識している程度で点滅していても問題ない。
- 高速に点灯→消灯を繰り返すようにプログラムを作成する。
- 16個を順番に点灯・消灯を繰り返すとすると、消費電力は、0.43Wだけ。
- つまり、瞬間的には1個のLEDだけがついている状態とする。

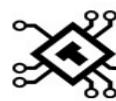




LEDのスタティック点灯・ダイナミ

- バーサライタ基板のLEDは、マトリクス接続されており、縦(ハイサイド)と横(ローサイド)の線が同時にonになると点灯する。
- フルカラー版ではさらに色を決めるRGBの信号線がある。
- 単色版では16～19行目がローサイド、21～24行目がハイサイドの制御線の駆動信号である。
- フルカラー版では28～31行目がローサイド、33～36行目がハイサイド、38～40行目がRGBの制御線の駆動信号である。
- 16行目で半固定抵抗のピンをアナログ入力として、potという変数名で定義している。
- 19行目がLEDを単位時間毎に点灯・消灯させるTickerクラスである。ここでは、led_blinkerという名前で定義している。

```
1. #include "mbed.h"
2. // #define USE_SPI // USE_I2C
3. #define COL_BK 0 // 000 BLACK
4. #define COL_BL 1 // 001 BLUE
5. #define COL_RD 2 // 010 RED
6. #define COL_PR 3 // 011 PURPLE
7. #define COL_GR 4 // 100 GREEN
8. #define COL_SB 5 // 101 SKY BLUE
9. #define COL_YL 6 // 110 YELLOW
10. #define COL_WH 7 // 111 WHITE
11.
12. #ifdef USE_SPI
13.     SPI gyro(dp2, dp1, dp6);
14.     DigitalOut scs(dp25);
15.
16.     DigitalOut led_l4(dp9);
17.     DigitalOut led_l3(dp10);
18.     DigitalOut led_l2(dp11);
19.     DigitalOut led_l1(dp18);
20.
21.     DigitalOut led_h4(dp26);
22.     DigitalOut led_h3(dp27);
23.     DigitalOut led_h2(dp5);
24.     DigitalOut led_h1(dp4);
25. #else
26.     I2C gyro(dp5, dp27);
27.
28.     DigitalOut led_l4(dp11);
29.     DigitalOut led_l3(dp10);
30.     DigitalOut led_l2(dp9);
31.     DigitalOut led_l1(dp4);
32.
33.     DigitalOut led_h4(dp26);
34.     DigitalOut led_h3(dp25);
35.     DigitalOut led_h2(dp18);
36.     DigitalOut led_h1(dp17);
37.
38.     DigitalOut led_nr(dp24);
39.     DigitalOut led_ng(dp2);
40.     DigitalOut led_nb(dp1);
41. #endif
```

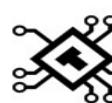


LEDのスタティック点灯・ダイナミ

- 45行目で半固定抵抗のピンをアナログ入力として, potという変数名で定義している.
- 47行目がLEDを単位時間毎に点灯・消灯させるTickerクラスである. ここでは, led_blinkerという名前で定義している.
- 52~77行目が, RGBの駆動信号の処理を行っている.

```
40.  DigitalOut led_nb(dp1);
41. #endif
42.
43. DigitalOut myled1(LED1);
44. DigitalOut myled2(LED2);
45. AnalogIn pot(dp13);
46. Serial pc(dp16, dp15); // tx, rx
47. Ticker led_blinker ;
48.
49. uint32_t tick = 0 ;
50. uint8_t on_led[16]={1,1,1,1, 1,1,1,1, 1,1,1,1, 1,1,1,1};
51.
52. void led_col(uint8_t num)
53. {
54. #ifdef USE_SPI
55.     return ;
56. #else
57.     switch(num){
58.         case COL_BK : led_ng = 1;led_nr = 1;led_nb = 1; // Black
59.                     break;
60.         case COL_BL : led_ng = 1;led_nr = 1;led_nb = 0; // blue
61.                     break;
62.         case COL_RD : led_ng = 1;led_nr = 0;led_nb = 1; // red
63.                     break;
64.         case COL_PR : led_ng = 1;led_nr = 0;led_nb = 0; // purple
65.                     break;
66.         case COL_GR : led_ng = 0;led_nr = 1;led_nb = 1; // green
67.                     break;
68.         case COL_SB : led_ng = 0;led_nr = 1;led_nb = 0; // sky blue
69.                     break;
70.         case COL_YL : led_ng = 0;led_nr = 0;led_nb = 1; // yellow
71.                     break;
72.         case COL_WH : led_ng = 0;led_nr = 0;led_nb = 0; // white
73.                     break;
74.         default :   led_ng = 1;led_nr = 1;led_nb = 1; // Black
75.     }
76. #endif
77. }

78. void led_on(uint8_t num,uint8_t col)
79. {
80.     led_nb = col;
81.     myled1 = col;
82.     myled2 = col;
83. }
```



LEDのスタティック点灯・ダイナミック点灯

- 79行目から、引数に1～16の数字と色番号を与えると、その数字に対応するLED(DS1～DS16)が点灯する関数である。
- 引数の数字に従って、ローサイド・ハイサイド・RGBをON/OFFしている。
- 引数に0を代入して呼び出すと、消灯する。
- 88～106行目が単色版のLEDの駆動信号の処理を行っている。

```
75.  }
76. #endif
77. }
78.
79. void led_on(uint8_t num,uint8_t col)
80. {
81.     led_h4=0;led_h3=0;led_h2=0;led_h1=0;led_l4=0;led_l3=0;led_l2=0;led_l1=0; led_col(COL_BK);
82.     if(num<17){
83.         if(col>0){
84.             led_col(col);
85.             switch(num){
86.                 case 0:led_l4=0;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=0;break;
87. #ifdef USE_SPI
88.                 case 1:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
89.                 case 2:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
90.                 case 3:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
91.                 case 4:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
92.
93.                 case 5:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
94.                 case 6:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
95.                 case 7:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
96.                 case 8:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
97.
98.                 case 9:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
99.                 case 10:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
100.                case 11:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
101.                case 12:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
102.
103.                case 13:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
104.                case 14:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
105.                case 15:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
106.                case 16:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
107.#else
108.    case 1:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
109.    case 2:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
110.    case 3:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
111.    case 4:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
112.
113.    case 5:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
114.    case 6:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
115.    case 7:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
```



LEDのスタティック点灯・ダイナミック点灯

- 108~127行目がフルカラー版のLEDの駆動信号の処理を行っている。
- 134~141行目がTickerクラスに与える周期実行される関数の定義である。
- 16個のLEDを順番に点灯するために、ledcnt++で1ずつ値を増やしつつ、led_on関数を呼び出している。
- led_cntが0~15の間の数字をとるように139行目で、16で割ったあまりをledcntに代入している。

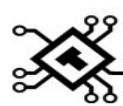
```
106.     case 16:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
107.#else
108.     case 1:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
109.     case 2:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
110.     case 3:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
111.     case 4:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
112.
113.     case 5:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
114.     case 6:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
115.     case 7:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
116.     case 8:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
117.
118.     case 9:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
119.     case 10:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
120.     case 11:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
121.     case 12:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
122.
123.     case 13:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
124.     case 14:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
125.     case 15:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
126.     case 16:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
127.#endif
128.     default:led_l4=0;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=0;
129.   }
130. }
131. }
132.}
133.
134.void led_blink_int(void)
135.{
136. static uint8_t ledcnt = 0 ;
137.
138. ledcnt++;
139. ledcnt = ledcnt % 16;
140. led_on(ledcnt+1,on_led[ledcnt]);
141.}
142.
143.int main(void)
144.{
145. int i = 0 ;
146. float pnd .
```



LEDのスタティック点灯・ダイナミック点灯

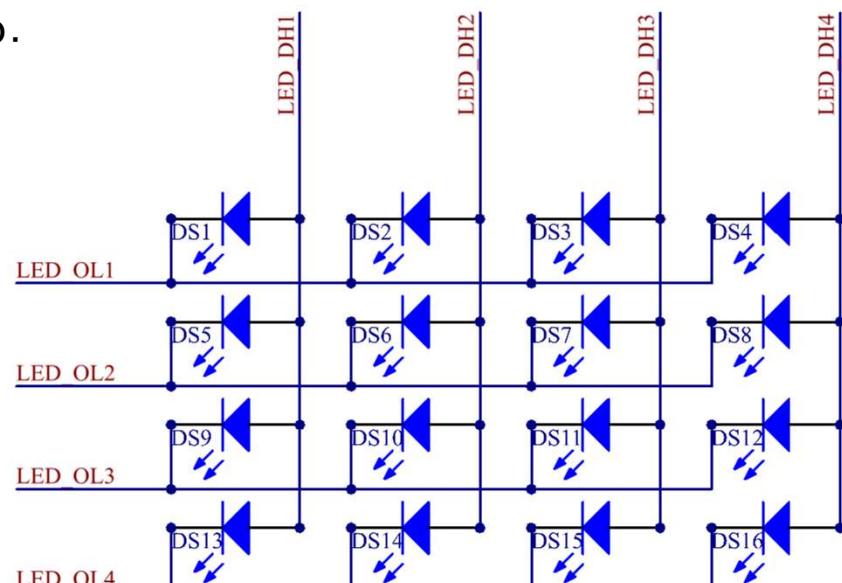
- 143行目からのmain関数では、これまでのプログラム同様のシリアルなどの設定に加えて、154行目で、Tickerクラスの設定を行っている。
- 151行目で取得したADコンバータの値をpotという変数に代入し、led_blink_intが実行される周期を設定する。
- 151行目の戻り値は、0.0～1.0間なので、154行目では、最大でも0.11秒までの周期しか設定できない。
- 目の残像を利用するためには、光の点滅は早いほど良い。
- しかし、使用マイコンのクロックにより最短の周期については制限がかかる。

```
137.
138. ledcnt++;
139. ledcnt = ledcnt % 16;
140. led_on(ledcnt+1,on_led[ledcnt]);
141.}
142.
143.int main(void)
144.{
145. int l = 0 ;
146. float prd ;
147.
148. pc.baud(115200);
149. pc.printf("LPC1114 Light Stick Demo\r\n");
150.
151. prd = pot.read();
152. pc.printf("prd = %f %7d \r\n",prd,(uint16_t)(prd*10000.0));
153.
154. led_blinker.attach_us(&led_blink_int,(uint16_t)(prd*10000.0)+100.0);
155.
156. myled1 = 1;
157. myled2 = 0;
158.
159. while(1){
160.     pc.printf("tick = %ld \r",tick);
161.     on_led[i] = 0 ;
162.     myled1 = myled1^1;
163.     myled2 = myled1^1;
164.     pc.printf("onled = %d \r\n",
165.             on_led[0],on_led[1],on_led[2],on_led[3],on_led[4],on_led[5],on_led[6],on_led[7],on_led[8],on_led[9],
166.             on_led[10],on_led[11],on_led[12],on_led[13],on_led[14],on_led[15]);
167.     wait(0.2);
168.     on_led[i] = 1 ;
169.     i++;
170.     i=i%16;
171. }
```



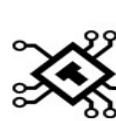
LEDのスタティック点灯・ダイナミック点灯

- 16回に一回だと、ちらつきが目立ち、かつ明るくない。
- これは、LED一個当たりわずか6%程度時間しか点灯していないためである。
- 消費電力は小さいが、ちらつきが大きい。
- 消費電力を抑えつつ、ちらつきも抑えることから、ある程度まとめて点灯する方式を考える。
- マトリクス接続を考慮しても、4個単位で制御するのが望ましい。
- そこで、4個単位でLEDを制御出来るように、関数を作る。
- 16個のLEDの状態を保存する変数を作る。
- その変数に基づき、ハイサイド/ローサイドを選択して、点灯を制御する。



マトリクス接続LED

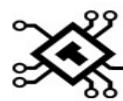
Nippon Institute of Technology



LEDのスタティック点灯・ダイ

- 右は改良したled_onである。
- LEDの1, 5, 9, 13は、ハイサイドを共有しているため、ハイサイドh1の信号を駆動する。
- 10~18行目が1, 5, 9, 13の点灯を制御している。

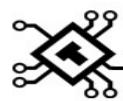
```
1. void led_on(uint8_t num,uint8_t col)
2. {
3.     led_h4=0;led_h3=0;led_h2=0;led_h1=0;led_l4=0;led_l3=0;led_l2=0;led_l1=0; led_col(COL_BK);
4.     if(num<17){
5.         if(col>0){
6.             led_col(col);
7.             switch(num){
8.                 case 0:led_l4=0;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=0;break;
9. #ifdef USE_SPI
10.             case 1:
11.             case 5:
12.             case 9:
13.                 case 13:led_l1 = (on_led[ 0])?1:0;
14.                     led_l2 = (on_led[ 4])?1:0;
15.                     led_l3 = (on_led[ 8])?1:0;
16.                     led_l4 = (on_led[12])?1:0;
17.                     led_h4=0;led_h3=0;led_h2=0;led_h1=1;
18.                     break ;
19.             case 2:
20.             case 6:
21.             case 10:
22.                 case 14:led_l1 = (on_led[ 1])?1:0;
23.                     led_l2 = (on_led[ 5])?1:0;
24.                     led_l3 = (on_led[ 9])?1:0;
25.                     led_l4 = (on_led[13])?1:0;
26.                     led_h4=0;led_h3=0;led_h2=1;led_h1=0;
27.                     break ;
28.             case 3:
29.             case 7:
30.             case 11:
31.                 case 15:led_l1 = (on_led[ 2])?1:0;
32.                     led_l2 = (on_led[ 6])?1:0;
33.                     led_l3 = (on_led[10])?1:0;
34.                     led_l4 = (on_led[14])?1:0;
35.                     led_h4=0;led_h3=1;led_h2=0;led_h1=0;
36.                     break ;
37.
38.             case 4:
39.             case 8:
40.             case 12:
41.                 case 16:led_l1 = (on_led[ 3])?1:0;
```



LEDのスタティック点灯・ダイ

- 右は改良したled_onである。
- LEDの1, 5, 9, 13は、ハイサイドを共有しているため、ハイサイドh1の信号を駆動する。
- 10~18行目が1, 5, 9, 13の点灯を制御している。
- それ以降、4本のピンをセットとして、LEDの点灯を制御していく。

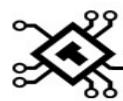
```
10.    case 1:
11.    case 5:
12.    case 9:
13.    case 13:led_l1 = (on_led[ 0])?1:0;
14.        led_l2 = (on_led[ 4])?1:0;
15.        led_l3 = (on_led[ 8])?1:0;
16.        led_l4 = (on_led[12])?1:0;
17.        led_h4=0;led_h3=0;led_h2=0;led_h1=1;
18.        break ;
19.    case 2:
20.    case 6:
21.    case 10:
22.    case 14:led_l1 = (on_led[ 1])?1:0;
23.        led_l2 = (on_led[ 5])?1:0;
24.        led_l3 = (on_led[ 9])?1:0;
25.        led_l4 = (on_led[13])?1:0;
26.        led_h4=0;led_h3=0;led_h2=1;led_h1=0;
27.        break ;
28.    case 3:
29.    case 7:
30.    case 11:
31.    case 15:led_l1 = (on_led[ 2])?1:0;
32.        led_l2 = (on_led[ 6])?1:0;
33.        led_l3 = (on_led[10])?1:0;
34.        led_l4 = (on_led[14])?1:0;
35.        led_h4=0;led_h3=1;led_h2=0;led_h1=0;
36.        break ;
37.
38.    case 4:
39.    case 8:
40.    case 12:
41.    case 16:led_l1 = (on_led[ 3])?1:0;
42.        led_l2 = (on_led[ 7])?1:0;
43.        led_l3 = (on_led[11])?1:0;
44.        led_l4 = (on_led[15])?1:0;
45.        led_h4=1;led_h3=0;led_h2=0;led_h1=0;
46.        break ;
47.    #else
48.        case 1:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
49.        case 2:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
50.        case 3:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
```



LEDのスタティック点灯・ダイ

- 右は改良したled_onである。
- LEDの1, 5, 9, 13は、ハイサイドを共有しているため、ハイサイドh1の信号を駆動する。
- 10~18行目が1, 5, 9, 13の点灯を制御している。
- それ以降、4本のピンをセットとして、LEDの点灯を制御していく。

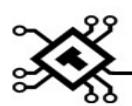
```
10.    case 1:
11.    case 5:
12.    case 9:
13.    case 13:led_l1 = (on_led[ 0])?1:0;
14.        led_l2 = (on_led[ 4])?1:0;
15.        led_l3 = (on_led[ 8])?1:0;
16.        led_l4 = (on_led[12])?1:0;
17.        led_h4=0;led_h3=0;led_h2=0;led_h1=1;
18.        break ;
19.    case 2:
20.    case 6:
21.    case 10:
22.    case 14:led_l1 = (on_led[ 1])?1:0;
23.        led_l2 = (on_led[ 5])?1:0;
24.        led_l3 = (on_led[ 9])?1:0;
25.        led_l4 = (on_led[13])?1:0;
26.        led_h4=0;led_h3=0;led_h2=1;led_h1=0;
27.        break ;
28.    case 3:
29.    case 7:
30.    case 11:
31.    case 15:led_l1 = (on_led[ 2])?1:0;
32.        led_l2 = (on_led[ 6])?1:0;
33.        led_l3 = (on_led[10])?1:0;
34.        led_l4 = (on_led[14])?1:0;
35.        led_h4=0;led_h3=1;led_h2=0;led_h1=0;
36.        break ;
37.
38.    case 4:
39.    case 8:
40.    case 12:
41.    case 16:led_l1 = (on_led[ 3])?1:0;
42.        led_l2 = (on_led[ 7])?1:0;
43.        led_l3 = (on_led[11])?1:0;
44.        led_l4 = (on_led[15])?1:0;
45.        led_h4=1;led_h3=0;led_h2=0;led_h1=0;
46.        break ;
47.    #else
48.        case 1:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
49.        case 2:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
50.        case 3:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
```



LEDのスタティック点灯・ダイナミック点灯(課題)

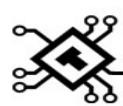
- 0～65535までの数字を2進数で点灯するプログラムを作成せよ。
- Ticker, Timerなどを駆使して、以下のビデオのような動きのあるLEDの点灯を実現せよ。





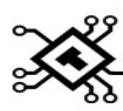
目次

- 1. 概要
- 2. 組み立て
- 3. mbed登録
- 4. Lチカに挑戦
- 5. タイマをつかう
- 6. センサを読み込む
- 7. センサとタイマを使った角度計算
- 8. LEDのスタティック点灯・ダイナミック点灯
- 9. 文字データの表示
- 10. 好きな文字を表示させてみよう



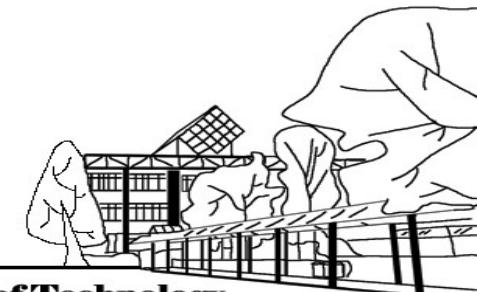
文字データの表示

- ここまでで、角度を計算する・LEDを点灯するが出来るようになった。
- そこで、特定の角度で、特定のパターンでLEDを光らせるバーサライタを完成させる。
- ここでは、例として、「日本工業大学[校章]」を使用する。
- 基本的なアルゴリズムは下記の2点である。
 - 表示させる文字列をLEDの点灯パターンにして配列に格納する。
 - ジャイロから計算される角度に応じて、配列の要素を取り出し、LEDで表示する。
- リストを次のスライドに示す。



CharaData.h

- 色番号でLEDの点灯を支持している.
 - なお配列幅は、16×120である.





```
1. #include "mbed.h"
2. #include "MPU9250.h"
3. #include "CharaData.h"
4. #define D2D_MPU9250 (0.06097561)
5. // #define USE_SPI // USE_I2C
6. #define RET_RANGE (-429) // -30 dps
7.
8. #define COL_BK 0 // 000 BLACK
9. #define COL_BL 1 // 001 BLUE
10. #define COL_RD 2 // 010 RED
11. #define COL_PR 3 // 011 PURPLE
12. #define COL_GR 4 // 100 GREEN
13. #define COL_SB 5 // 101 SKY BLUE
14. #define COL_YL 6 // 110 YELLOW
15. #define COL_WH 7 // 111 WHITE
16.
17. #ifdef USE_SPI
18.     SPI gyro(dp2, dp1, dp6);
19.     DigitalOut scs(dp25);
20.
21.     DigitalOut led_l4(dp9);
22.     DigitalOut led_l3(dp10);
23.     DigitalOut led_l2(dp11);
24.     DigitalOut led_l1(dp18);
25.
26.     DigitalOut led_h4(dp26);
27.     DigitalOut led_h3(dp27);
28.     DigitalOut led_h2(dp5);
29.     DigitalOut led_h1(dp4);
30. #else
31.     I2C gyro(dp5, dp27);
32.
33.     DigitalOut led_l4(dp11);
34.     DigitalOut led_l3(dp10);
35.     DigitalOut led_l2(dp9);
36.     DigitalOut led_l1(dp4);
37.
38.     DigitalOut led_h4(dp26);
39.
40.     DigitalOut led_l3(dp10);
41.     DigitalOut led_l2(dp9);
42.     DigitalOut led_l1(dp4);
43.
44.     DigitalOut led_h4(dp26);
45.     DigitalOut led_h3(dp25);
46.     DigitalOut led_h2(dp18);
47.     DigitalOut led_h1(dp17);
48.
49.     DigitalOut led_nr(dp24);
50.     DigitalOut led_ng(dp2);
51.     DigitalOut led_nb(dp1);
52.
53. #endif
54.     DigitalOut myled1(LED1);
55.     DigitalOut myled2(LED2);
56.     AnalogIn pot(dp13);
57.     Serial pc(dp16, dp15); // tx, rx
58.     Ticker gyro_handler;
59.     Ticker led_blinker ;
60.
61.     uint8_t gyro_i2c_addr = MPU9250_I2C_ADDR;
62.     uint8_t gyro_l_addr = MPU9250_GYRO_ZOUT_L, gyro_h_addr = MPU9250_GYRO_ZOUT_H;
63.     uint8_t gyro_read_flg = MPU9250_READ, gyro_write_flg = MPU9250_WRITE;
64.     float dig2dps = D2D_MPU9250;
65.     float angle = 0.0;
66.     float z_bias;
67.     uint32_t tick = 0 ;
68.     uint8_t on_led[16]={1,1,1,1, 1,1,1,1, 1,1,1,1, 1,1,1,1};
69.
70. #ifdef USE_SPI
71.     uint8_t gyro_read_write(uint8_t addr,uint8_t dat)
72.     {
73.         uint8_t ret;
74.         scs = 1;
75.         scs = 0 ;
76.         gyro.write(addr);
77.         ret = gyro.write(dat);
78.         scs = 1;
79.         return ret;
80.     }
81.
82.     void gyro_write(uint8_t addr,uint8_t dat)
83.
```



```
75. void gyro_write(uint8_t addr,uint8_t dat)
76. {
77.   gyro_read_write(addr|gyro_write_flg,dat);
78. }
79.
80. uint8_t gyro_read(uint8_t addr)
81. {
82.   uint8_t ret ;
83.   ret = gyro_read_write(addr|gyro_read_flg,0x00);
84.   return ret;
85. }
86. #else
87. void gyro_write(uint8_t addr,uint8_t dat)
88. {
89.   uint8_t sdat[2];
90.   sdat[0] = addr ;
91.   sdat[1] = dat ;
92.   gyro.write((int)gyro_i2c_addr<<1,(char*)sdat, 2, 0); // stop
93. }
94.
95. uint8_t gyro_read(uint8_t addr)
96. {
97.   uint8_t ret ;
98.   gyro.write((int)gyro_i2c_addr<<1,(char*)&addr, 1, 1); // no stop
99.   gyro.read((int)gyro_i2c_addr<<1,(char*)&ret, 1, 0);
100.  return ret;
101. }
102.#endif
103.
104.void gyro_bias(void)
105.{
106.  uint16_t i;
107.  uint16_t tz_data;
108.  int32_t z_data;
109.  uint8_t dmy;
110.
111.  z_bias = 0.0;
112.
```

```
108.  int32_t z_data;
109.  uint8_t dmy;
110.
111.  z_bias = 0.0;
112.
113.  for(i=0;i<32;i++){
114.    dmy=gyro_read(gyro_l_addr);
115.    tz_data = (0x00ff&dmy);
116.    dmy=gyro_read(gyro_h_addr);
117.    tz_data |= (0xff00&(dmy<<8));
118.    z_data = -1*(int16_t)tz_data;
119.    z_bias += (float)z_data*dig2dps;
120.    wait_ms(10);
121.  }
122.  z_bias = z_bias / 32;
123.}
124.
125.void gyro_init(void)
126.{
127.  int ret ;
128.  uint8_t mpu_adr = 0;
129.  uint8_t mpu_reg = 0;
130.
131.  myled1 = myled2 = 0;
132.
133.#ifdef USE_SPI
134.  gyro.format(8,3);
135.  gyro.frequency(5000000);
136.  scs = 1;
137.#else
138.  gyro.frequency(400000);
139.#endif
140.
141.  ret = gyro_read(MPU9250_WHO_AM_I);
142.  pc.printf("WHOAMI register = 0x%X \n\r", ret);
143.  if(ret == I_AM_MPU9250) {
144.    pc.printf("MPU9250 is found \n\r");
145.    myled1 = 1;
146.  }else{
147.    pc.printf("no MPU9250 is found \n\r");
148.    return ;
149.  }
150.
```



```
151.do{
152.     mpu_adr = MPU9250_PWR_MGMT_1 ;
153.     mpu_reg = MPU9250_H_RESET ;
154.     gyro_write(mpu_adr,mpu_reg);
155.     wait_ms(10);
156.
157.     mpu_adr = MPU9250_PWR_MGMT_1 ;
158.     mpu_reg = 0x00 ;
159.     gyro_write(mpu_adr,mpu_reg);
160.
161.     mpu_adr = MPU9250_PWR_MGMT_1 ;
162.     ret=gyro_read(mpu_adr);
163. }while(ret != mpu_reg);
164.
165. gyro_write(MPU9250_CONFIG,MPI9250_GBW_250);
166. gyro_write(MPU9250_SMPLRT_DIV, 0x00);
167.
168. do{
169.     mpu_adr = MPU9250_GYRO_CONFIG ;
170.     ret = gyro_read(mpu_adr);
171.     ret = ret & ~0x03; // Clear Fchoice bits [1:0]
172.     ret = ret & ~0x18; // Clear FS bits [4:3]
173.     mpu_reg = ret | MPI9250_GFS_2000; // Set full scale range for the gyro
174.     gyro_write(mpu_adr,mpu_reg);
175.     ret = gyro_read(mpu_adr);
176. }while(ret != mpu_reg);
177.
178. myled2 = 1;
179. pc.printf("I did it \n\r");
180.}
181.
182. void gyro_int(void)
183.{
184.     static uint16_t tz_data[4]={0,0,0,0};
185.     int32_t z_data;
186.     uint8_t dmy;
187.     int16_t i;
188.     uint16_t idx ;
```

```
183.     static uint16_t tz_data[4]={0,0,0,0};
184.     int32_t z_data;
185.     uint8_t dmy;
186.     int16_t i;
187.     uint16_t idx ;
188.
189.     for(i=2;i>=0;i--){
190.         tz_data[i+1] = tz_data[i] ;
191.     }
192.
193.     dmy=gyro_read(gyro_l_addr);
194.     tz_data[0] = (0x00ff&dmy);
195.     dmy=gyro_read(gyro_h_addr);
196.     tz_data[0] |= (0xff00&(dmy<<8));
197.     z_data = -1*(int16_t)tz_data[0];
198.     tz_data[0] = (uint16_t)z_data;
199.
200.     for(i=1;i<4;i++){
201.         z_data += (int16_t)tz_data[i];
202.     }
203.     z_data = z_data / 4;
204.
205.     angle += ((float)z_data*dig2dps-z_bias)*0.001;
206.     tick++;
207.
208.     if((int16_t)tz_data[0] < RET_RANGE)
209.         if((int16_t)tz_data[1] < RET_RANGE)
210.             if((int16_t)tz_data[2] < RET_RANGE)
211.                 if((int16_t)tz_data[3] < RET_RANGE)
212.                     angle = 0.0;
213.
214.     if((angle > (float)MOJISUU_OFS)&&(angle < (float)(MOJISUU+MOJISUU_OFS))){
215. //     idx = MOJISUU - (uint16_t)(angle - (float)MOJISUU_OFS);
216.     idx = (uint16_t)(angle - (float)MOJISUU_OFS);
217.
218.     if(idx>MOJISUU)
219.         idx=MOJISUU-1;
220.     for(i=0;i<16;i++)
221.         on_led[i]=moji[i][idx];
222. }else{
223.     for(i=0;i<16;i++)
224.         on_led[i]=0 ;
225. }
```



```
214.if((angle > (float)MOJISUU_OFS)&&(angle < (float)(MOJISUU+MOJISUU_OFS)) {
215.//    idx = MOJISUU - (uint16_t)(angle - (float)MOJISUU_OFS);
216.    idx = (uint16_t)(angle - (float)MOJISUU_OFS);
217.
218.    if(idx>MOJISUU)
219.        idx=MOJISUU-1;
220.    for(i=0;i<16;i++)
221.        on_led[i]=moji[i][idx];
222.    }else{
223.        for(i=0;i<16;i++)
224.            on_led[i]=0 ;
225.    }
226.}
227.
228.void led_col(uint8_t num)
229.{
230.#ifdef USE_SPI
231.    return ;
232.#else
233.    switch(num){
234.        case COL_BK : led_ng = 1;led_nr = 1;led_nb = 1; // Black
235.            break;
236.        case COL_BL : led_ng = 1;led_nr = 1;led_nb = 0; // blue
237.            break;
238.        case COL_RD : led_ng = 1;led_nr = 0;led_nb = 1; // red
239.            break;
240.        case COL_PR : led_ng = 1;led_nr = 0;led_nb = 0; // purple
241.            break;
242.        case COL_GR : led_ng = 0;led_nr = 1;led_nb = 1; // green
243.            break;
244.        case COL_SB : led_ng = 0;led_nr = 1;led_nb = 0; // sky blue
245.            break;
246.        case COL_YL : led_ng = 0;led_nr = 0;led_nb = 1; // yellow
247.            break;
248.        case COL_WH : led_ng = 0;led_nr = 0;led_nb = 0; // white
249.            break;
250.        default :    led_ng = 1;led_nr = 1;led_nb = 1; // Black
251.    }
}
```

```
247.            break;
248.        case COL_WH : led_ng = 0;led_nr = 0;led_nb = 0; // white
249.            break;
250.        default :    led_ng = 1;led_nr = 1;led_nb = 1; // Black
251.    }
252.#endif
253.
254.
255.void led_on(uint8_t num,uint8_t col)
256.{
257.    led_h4=0;led_h3=0;led_h2=0;led_h1=0;led_l4=0;led_l3=0;led_l2=0;led_l1=0; led_col(COL_BK);
258.    if(num<17){
259.        if(col>0){
260.            led_col(col);
261.
262.            switch(num){
263.                case 0:led_l4=0;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=0;break;
264.
265.#ifdef USE_SPI
266.                case 1:
267.                case 5:
268.                case 9:
269.                case 13:led_l1 = (on_led[ 0])?1:0;
270.                led_l2 = (on_led[ 4])?1:0;
271.                led_l3 = (on_led[ 8])?1:0;
272.                led_l4 = (on_led[12])?1:0;
273.                led_h4=0;led_h3=0;led_h2=0;led_h1=1;
274.                break ;
275.
276.                case 2:
277.                case 6:
278.                case 10:
279.                case 14:led_l1 = (on_led[ 1])?1:0;
280.                led_l2 = (on_led[ 5])?1:0;
281.                led_l3 = (on_led[ 9])?1:0;
282.                led_l4 = (on_led[13])?1:0;
283.                led_h4=0;led_h3=0;led_h2=1;led_h1=0;
284.                break ;
285.
286.                case 3:
287.                case 7:
288.                case 11:
289.                case 15:led_l1 = (on_led[ 2])?1:0;
```

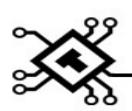


```
285.
286. case 3:
287. case 7:
288. case 11:
289. case 15:led_l1 = (on_led[ 2])?1:0;
290.     led_l2 = (on_led[ 6])?1:0;
291.     led_l3 = (on_led[10])?1:0;
292.     led_l4 = (on_led[14])?1:0;
293.     led_h4=0;led_h3=1;led_h2=0;led_h1=0;
294.     break ;
295.
296. case 4:
297. case 8:
298. case 12:
299. case 16:led_l1 = (on_led[ 3])?1:0;
300.     led_l2 = (on_led[ 7])?1:0;
301.     led_l3 = (on_led[11])?1:0;
302.     led_l4 = (on_led[15])?1:0;
303.     led_h4=1;led_h3=0;led_h2=0;led_h1=0;
304.     break ;
305.#else
306. case 1:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
307. case 2:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
308. case 3:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
309. case 4:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=1;break;
310.
311. case 5:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
312. case 6:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
313. case 7:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
314. case 8:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=1;led_h1=0;break;
315.
316. case 9:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
317. case 10:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
318. case 11:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
319. case 12:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=1;led_h2=0;led_h1=0;break;
320.
321. case 13:led_l4=0;led_l3=0;led_l2=0;led_l1=1;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
322. case 14:led_l4=0;led_l3=0;led_l2=1;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
323. case 15:led_l4=0;led_l3=1;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
324. case 16:led_l4=1;led_l3=0;led_l2=0;led_l1=0;led_h4=1;led_h3=0;led_h2=0;led_h1=0;break;
325.#endif
326.     default:led_l4=0;led_l3=0;led_l2=0;led_l1=0;led_h4=0;led_h3=0;led_h2=0;led_h1=0;
327.     }
328. }
329. }
330. }
331.
332. void led_blink_int(void)
333. {
334.     static uint8_t ledcnt = 0 ;
335.
336.     ledcnt++;
337.     ledcnt = ledcnt % 16;
338.     led_on(ledcnt+1,on_led[ledcnt]);
339. }
340.
341. int main(void)
342. {
343.     int32_t z_data;
344.     uint16_t tz_data;
345.     uint8_t dmy;
346.     float prd;
347.
348.     pc.baud(115200);
349.     pc.printf("LPC1114 Light Stick Demo\r\n");
350.
351.     gyro_init();
352.     gyro_bias();
353.     gyro_handler.attach_us(&gyro_int,1000);
354.
355.     prd = pot.read();
356.     pc.printf("prd = %lf %7d \n\r",prd,(uint16_t)(prd*10000.0));

```

```
340.
341.int main(void)
342.{  
343.    int32_t z_data;  
344.    uint16_t tz_data;  
345.    uint8_t dmy;  
346.    float prd ;  
347.  
348.    pc.baud(115200);  
349.    pc.printf("LPC1114 Light Stick Demo\r\n");  
350.  
351.    gyro_init();  
352.    gyro_bias();  
353.    gyro_handler.attach_us(&gyro_int,1000);  
  
354.    prd = pot.read();  
355.    pc.printf("prd = %ld \r",prd,(uint16_t)(prd*10000.0));  
356.    led_blinker.attach_us(&led_blink_int,(uint16_t)(prd*10000.0)+100.0);  
  
357.    myled1 = 1;  
358.    myled2 = 0;  
359.  
360.    while(1){  
361.#if 1  
362.        dmy=gyro_read(MPU9250_TEMP_OUT_L);  
363.        tz_data = (0x00ff&dmy);  
364.        dmy=gyro_read(MPU9250_TEMP_OUT_H);  
365.        tz_data |= (0xff00&(dmy<<8));  
366.        z_data = (int16_t)(tz_data);  
367.        pc.printf("temp = %f, ", (double)z_data/333.87+21.0);  
368.#endif  
369.        pc.printf("tick = %d, angle = %f \r",tick,angle);  
370.  
371.        myled1 = myled1^1;  
372.        myled2 = myled1^1;  
373.    }  
374.}
```





自分の文字を考える。

- CharaData.hを変更して好きな文字を表示させよう。