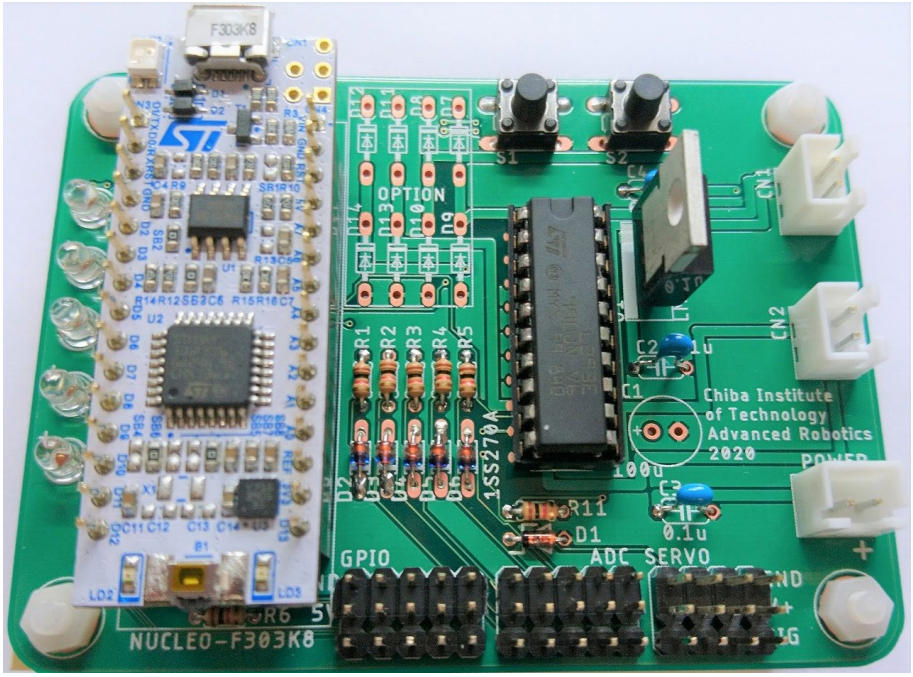
千葉工業大学

未来ロボティクス学科

実習用制御回路

スタートガイド



2024 年 4 月 21 日版

千葉工業大学

未来ロボティクス学科

目次

１．実習用制御回路に関して

１．１　未来ロボティクス学科実習用制御回路

未来ロボティクス学科では，学生実習用として独自の制御回路を開発して提供しています．本回路は，みなさんの身の回りで多く使用されている **ARM** で，自動制御を簡易的に体験できるように構成してあります．

ARM
スマートフォン(iPhone, Android)やゲーム機(Switch)など様々なところで使用されているCPUコアアーキテクチャ

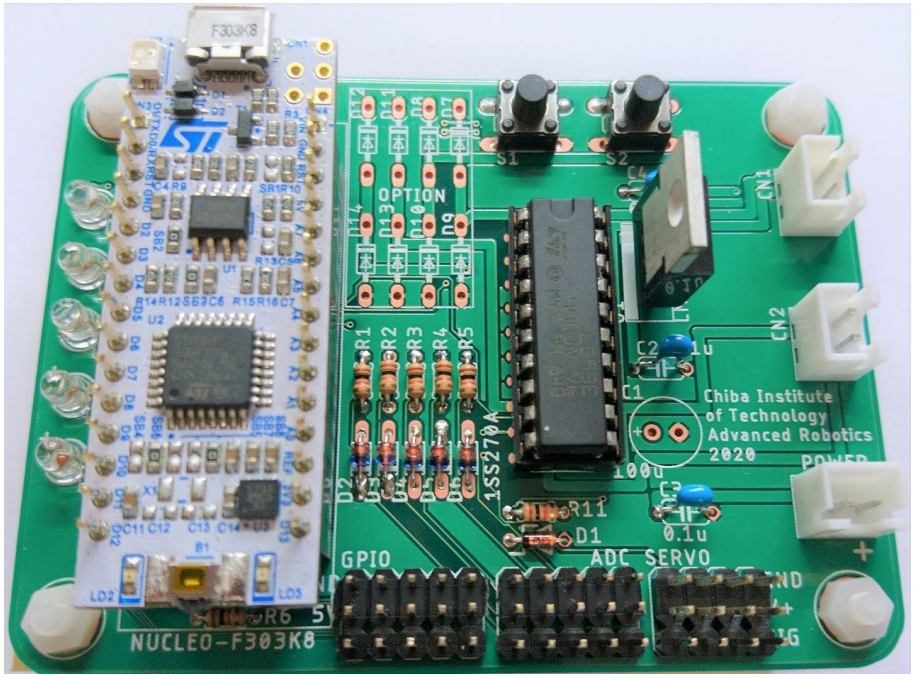


図　実習で使用する制御回路 ※

※モータドライバなど一部の部品を変更しています．

本制御回路は，主に以下の特徴があります．

・プログラムの開発が比較的簡単です．
　（mbed 標準の環境で開発できます．）
・様々な外部機器を接続して制御することができます．

接続できる機器は，主に以下の通りです．

・LED	５個（制御回路に搭載）
・スイッチ	２個（制御回路に搭載）
・ラジコン用サーボモータ	４個
・モータ	２個
・汎用入出力	５個（LED と共用）
・センサ（アナログ入力）	５個

１．２　mbed

mbed（エンベッド）とは ARM 社のプロトタイピング用ワンボードマイコンおよびそのデバイスのプログラミング環境を指します．多くのプロトタイプ（試作品）が mbed で制御されています．用途に応じて様々な種類がありますが，本実習ではその中でも比較的コンパクトなものを使用します．mbed を使用してロボットを開発することで，様々な電子機器を制御する基本を学ぶことができます．ウェブ上に資料やサンプルプログラムが多数ありますので，覗いてみるとよいでしょう．

mbed サイト
http://mbed.org/

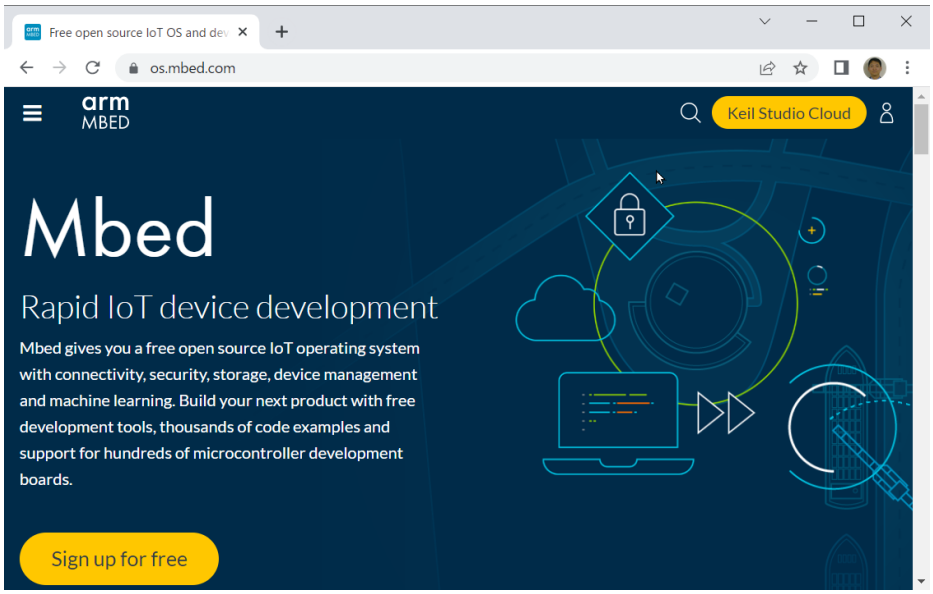


図　mbed のホーム画面

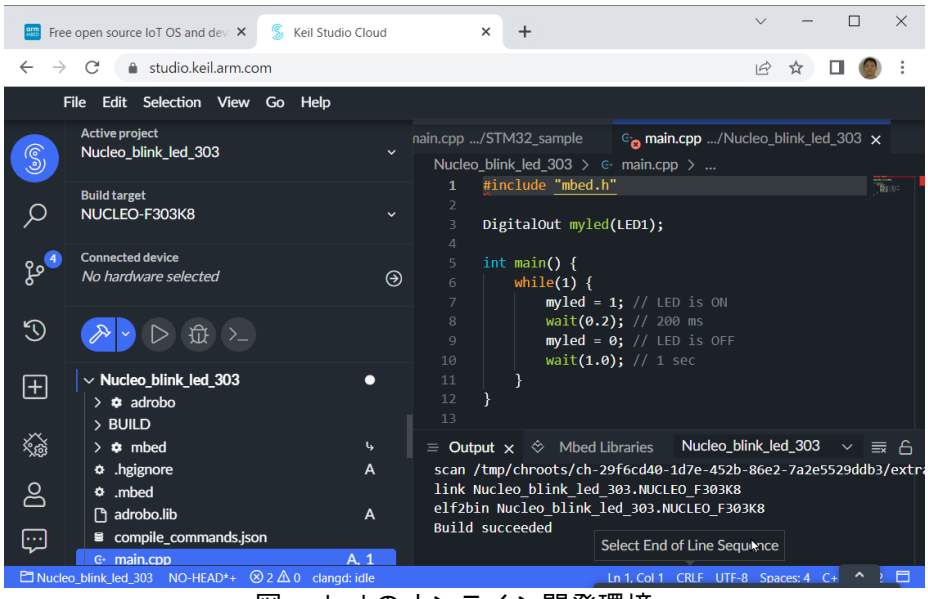
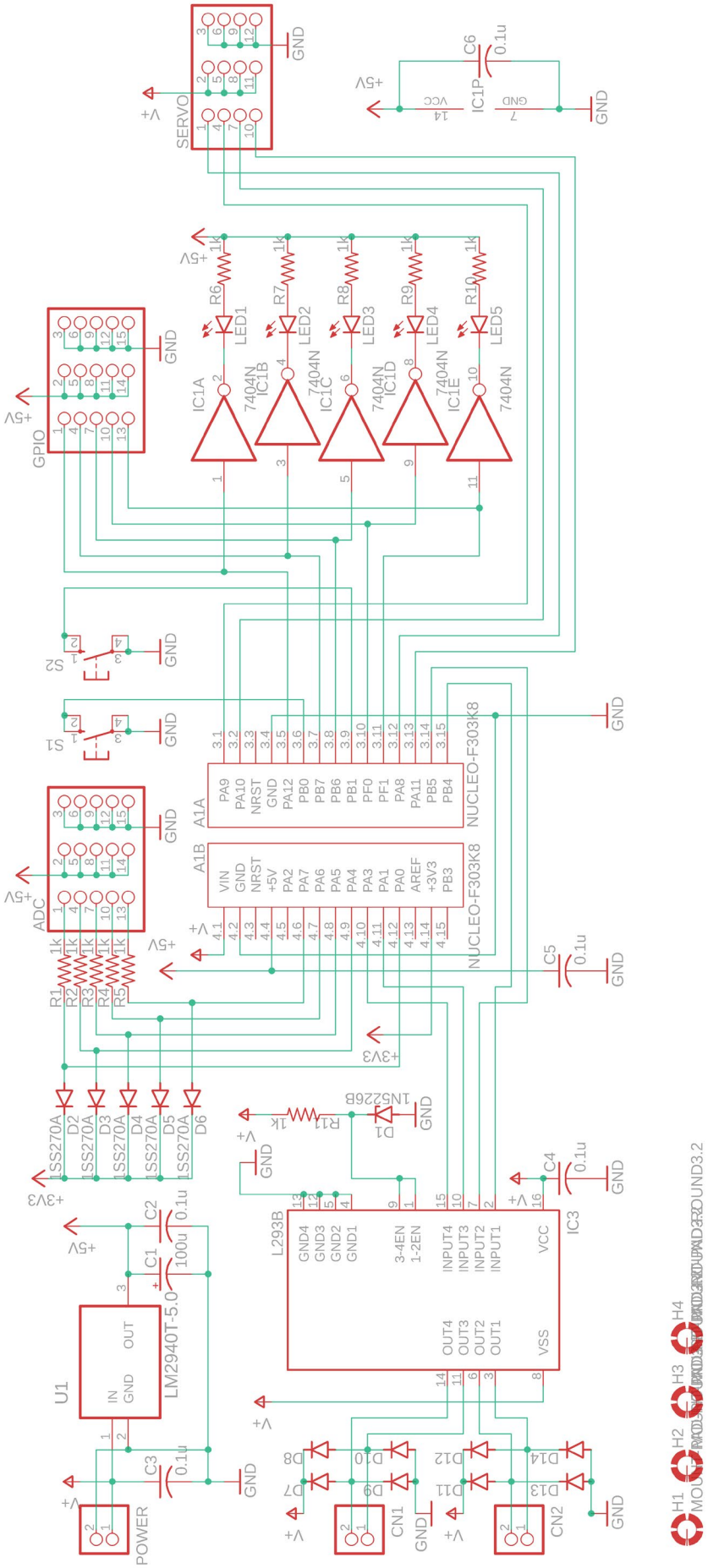


図　mbed のオンライン開発環境

１．３　制御回路の回路図


SERVO
GND
+5V
C6
0.1u
IC1P
VCC
GND
R6
R7
R8
R9
R10
1k
LED1
LED2
LED3
LED4
LED5
IC1A
IC1B
IC1C
IC1D
IC1E
7404N
GPIO
S2
S1
NUCLEO-F303K8
A1A
A1B
PA9
PA10
NRST
PA12
PB0
PB7
PB6
PB1
PF0
PF1
PA8
PA11
PB5
PB4
VIN
+5V
PA2
PA7
PA6
PA5
PA4
PA3
PA1
PA0
AREF
+3V3
PB3
ADC
R1
R2
R3
R4
R5
C5
+3V3
1SS270A
D2
D3
D4
D5
D6
1N5226B
D1
R11
C4
L293B
IC3
GND4
GND3
GND2
GND1
3-4EN
1-2EN
INPUT4
INPUT3
INPUT2
INPUT1
VCC
OUT4
OUT3
OUT2
OUT1
VSS
U1
LM2940T-5.0
IN
GND
OUT
C2
C1
100u
C3
POWER
D7
D8
D9
D10
D11
D12
D13
D14
CN1
CN2
H1
H2
H3
H4

１．４　ピン配置

ピン配置を示します．外部機器と接続する端子としては以下があります．

・モータ
・サーボモータ
・A/D 変換
・汎用 I/O(GPIO)

サーボモータ，A/D 変換器，GPIO の端子は GND,+電源の端子もセットになっており，例えばサーボモータは接続するだけで電源が供給され制御することができます．ここで，提供される+電源は以下となります．

１）電池の電圧　モータ，サーボモータ　　POWER に電池を接続しないと電源が供給されません．
２）＋5V　　A/D 変換，汎用 I/O(GPIO)　USB で接続するだけで電源が供給されます．

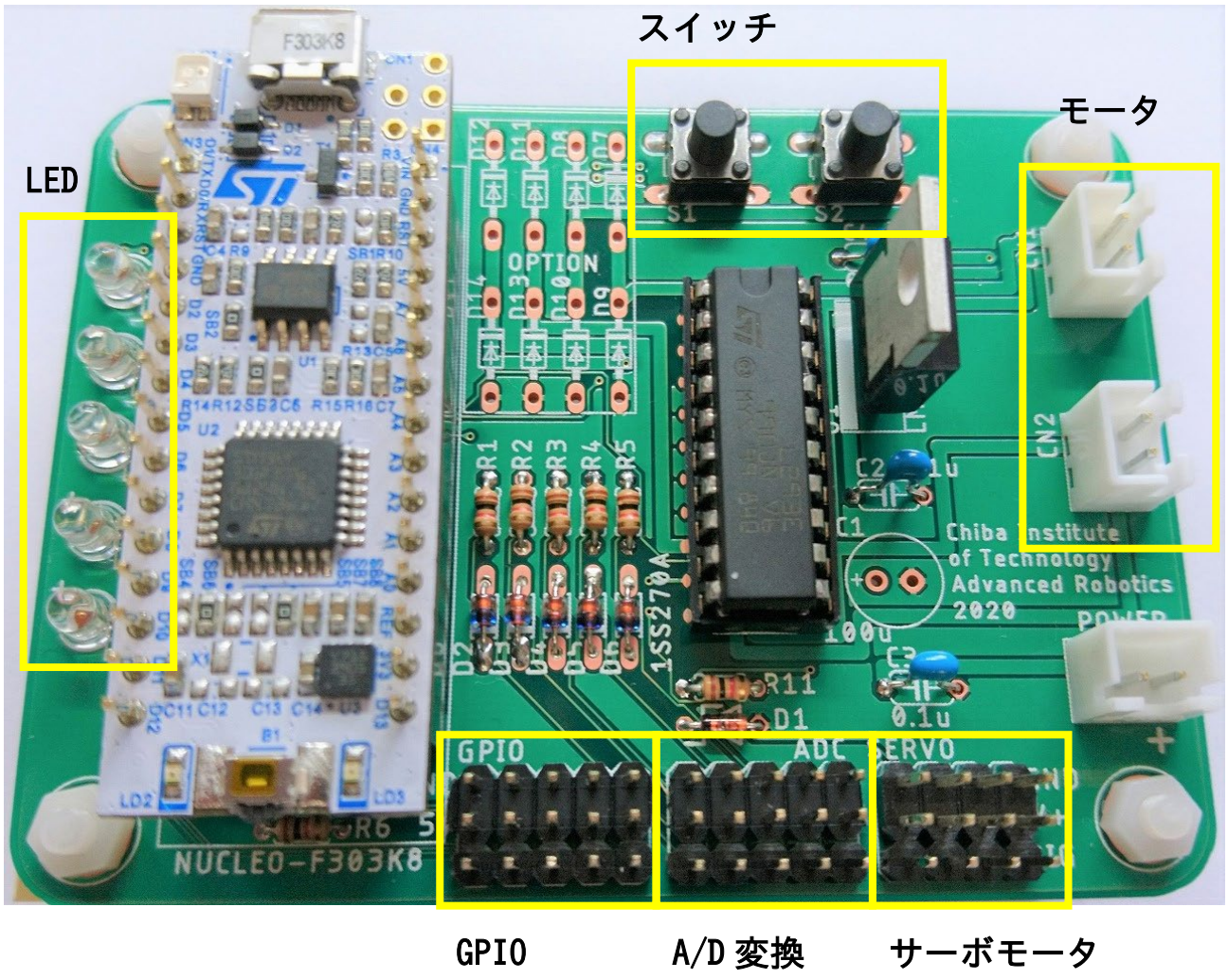


図　制御回路のピン配置
GPOIO，A/D 変換，サーボモータはそれぞれ左から１番

２．制御回路の製作に関して

２．１　制御回路の部品

制御回路は多くの部品から構成されます.以下に制御回路の部品を示します.これらの部品を所定の場所にはんだ付けしていくことが必要になります.

表　電子部品のリスト

名称	型番	個数	
コンピュータ	NUCLEO-F303K8	1	
モータドライバ	L293B	1	2024年度からL293Bに変更．以前はL293E
インバータ	TC74HC04APF	1	
レギュレータ	LM2940T-5.0	1	
汎用ダイオード	1SS270A	5	
ツェナーダイオード	1N5226B	1	
ファーストリカバリダイオード※	ERA22-04V3※	8	※オプション 大電流を流さない場合は不要です.
抵抗	1kΩ, 1/6W	11	
積層セラミックコンデンサ	0.1uF	5	
電解コンデンサ	100uF	1	
LED	OSG8HA3Z74A	5	
タクトスイッチ		2	
ピンヘッダ	PSS-430256-05	2	
ピンヘッダ	PSS-430256-04	1	
コネクタ	B2B-XH-A	3	
コネクタ	XHP-2	3	
コンタクト	SXH-001T-P0.6	6	
電池ボックス	SBH-341-1AS	1	
IC ソケット	16P	1	2024 年度から 20P->16P
IC ソケット	14P	1	
ピンソケット（メス）	1x15	2	
六角スペーサ	P-01864	4	
基板	自作	1	

２．２　はんだ付けに関して

はんだ付けに関しては，講義で解説があると思います．それをよく聞いて手際よく作るようにしてください．ダイオードや IC など半導体を使用している部品は加熱しすぎると壊れますので，気をつけるようにしてください．

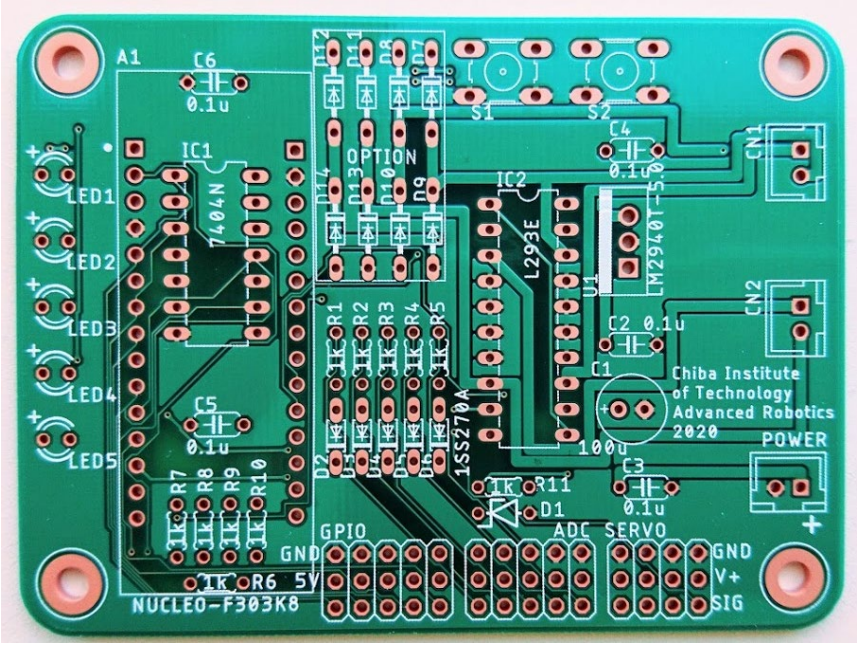
【適宜内容追加予定】

２．３　製作の手順

製作手順の一例を示します．必ずしもこの手順で製作しなくても構いません．なお，基本は低い部品から高い部品へとはんだ付けすると，比較的楽に作業が行なえます．

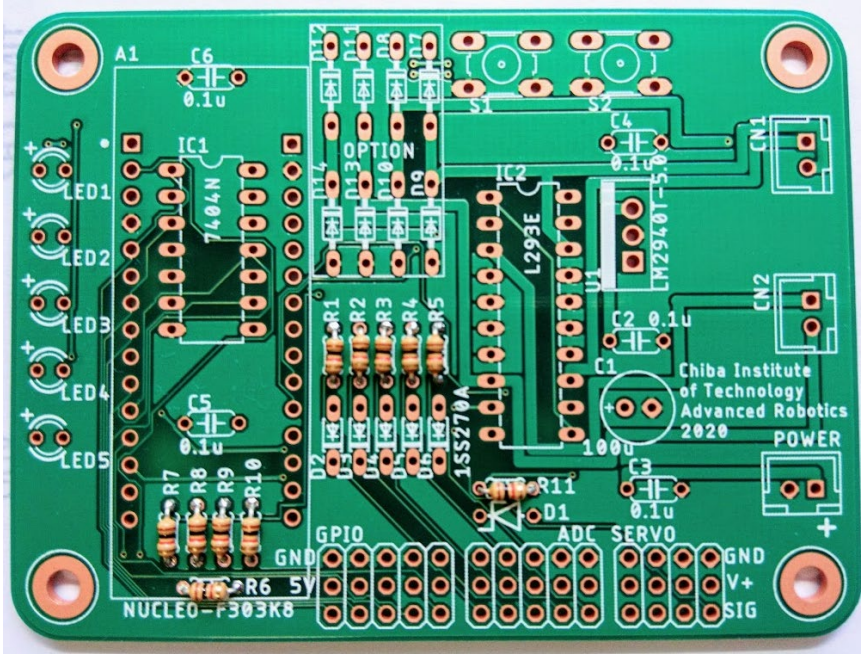
基板は 2024 年版に更新されていますが，作業は一緒です．
（パターンは一部変更されています．）

１）基板を確認
まずは基板をよく見てどこに何を取り付けるかを確認しましょう．



基板の記号は回路図と対応しています．どのように配線されているか知りたくなったら，回路図を見ましょう．

２）抵抗の取り付け
1kΩの抵抗を 11 個取り付けましょう．向きは関係ありません．

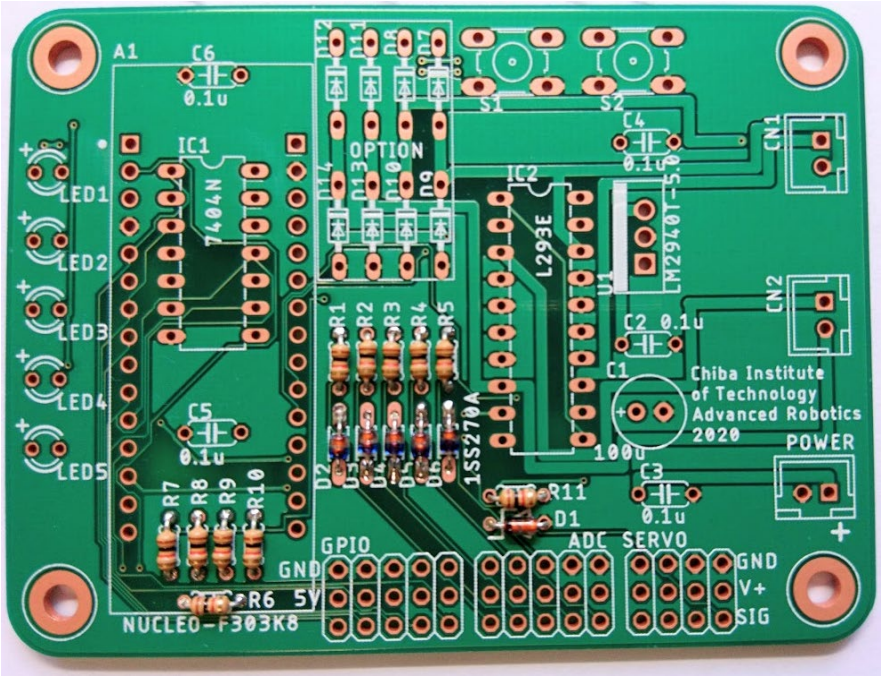


抵抗の端子を奥まで穴に差し込んだ後に少し曲げましょう．
曲げないとはんだ付けするときに，動いてしまうおそれがあります．

３）ダイオードの取り付け

汎用ダイオード５つ，**ツェナーダイオード１つ**を取り付けましょう．よく見て間違えないように取り付けましょう．また，**向きがあります**ので，逆向きに取り付けないでください．さらに，熱に弱いのでハンダゴテで加熱するのは５秒ぐらいにしてください．

最も注意しなければならないはんだ付け作業の１つです．
加熱しすぎると部品が故障します．



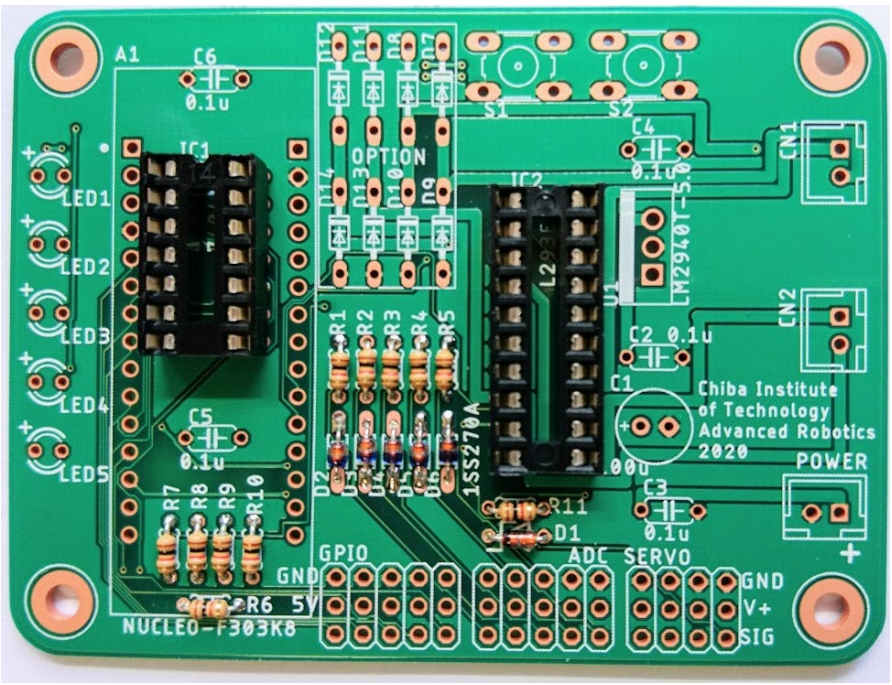
汎用ダイオードとツェナーダイオードの位置を間違えないようにしてください．

４）ソケットの取り付け

IC を取り付けるためのソケットをはんだ付けします．正しい向きに取り付けるようにしてください．浮かないように取り付けてください．

逆向きに取り付けても取り付ける IC の向きが間違っていなければ動作はします．

ただし，できるだけ正しい向きに取り付けてください．

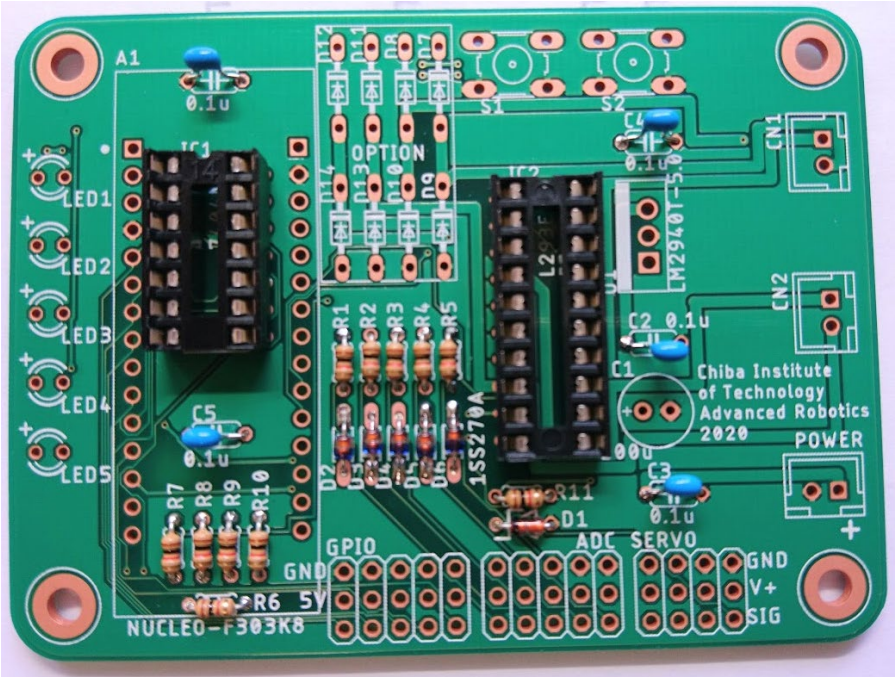


ソケットが基板から浮かないようにするためには，１つだけ足をはんだ付けして，押し付けながら再度ハンダを溶かすと隙間がなくなります．

右のソケットは **20P->16P に変更**(L293E->L293B に変更のため)

５）セラミックコンデンサの取り付け

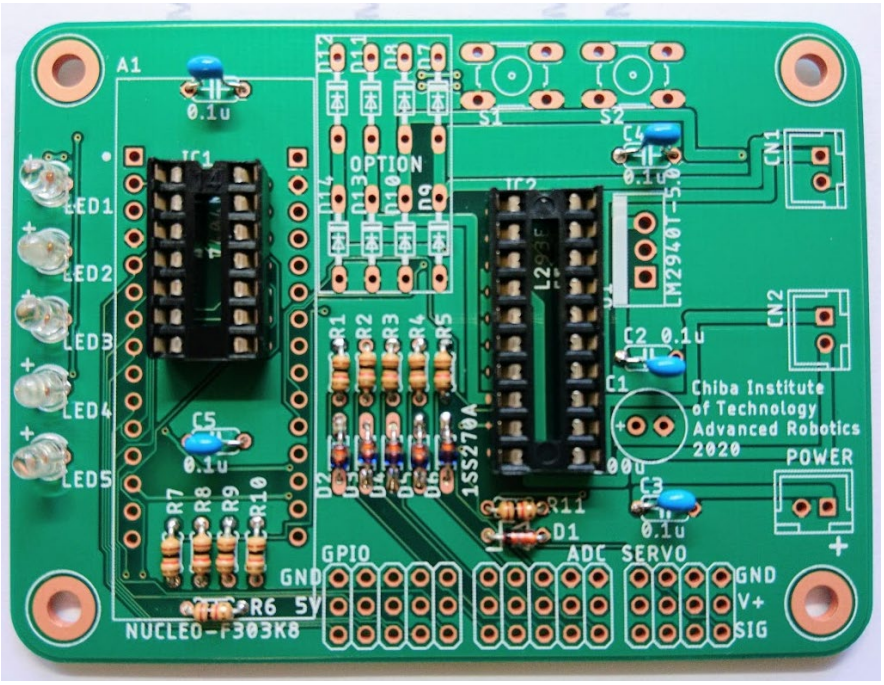
0.1μF のセラミックコンデンサを５つ取り付けてください．向きはありません．



６）LED の取り付け

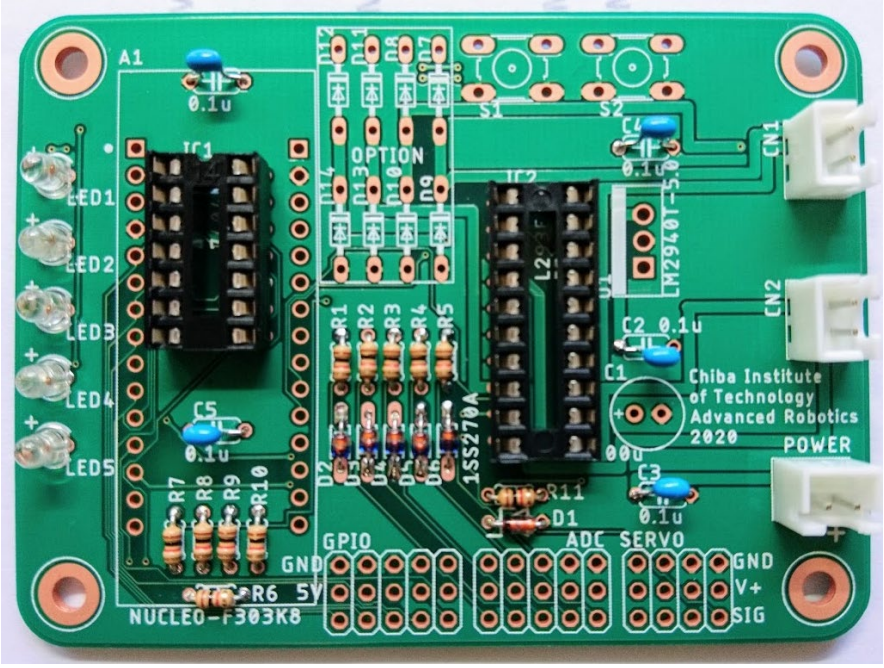
LED は半導体ですので，加熱は５秒程度にしてください．また，向きがありますので気をつけてください．足の長い方が＋です．

逆に取り付けると点灯しません．点灯しない以外の問題はありません．



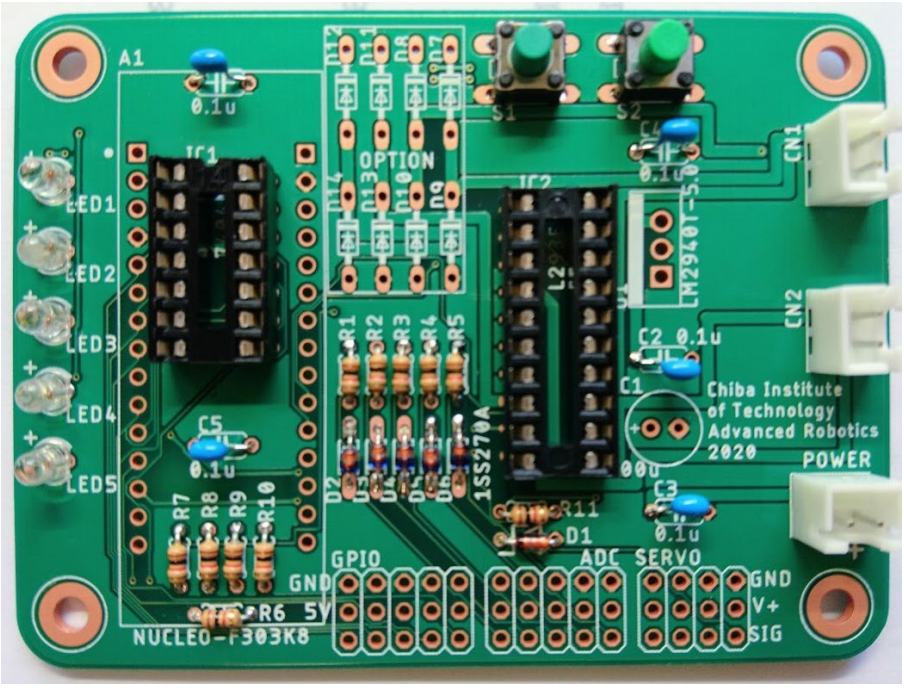
７）コネクタの取り付け

コネクタ（レセプタクル）を取り付けてください．向きがあります．また，浮かないように取り付けてください．



８）スイッチを取り付けます．

色はどれでも構いません．できるだけ奥まで差し込んでから，はんだ付けしてください．

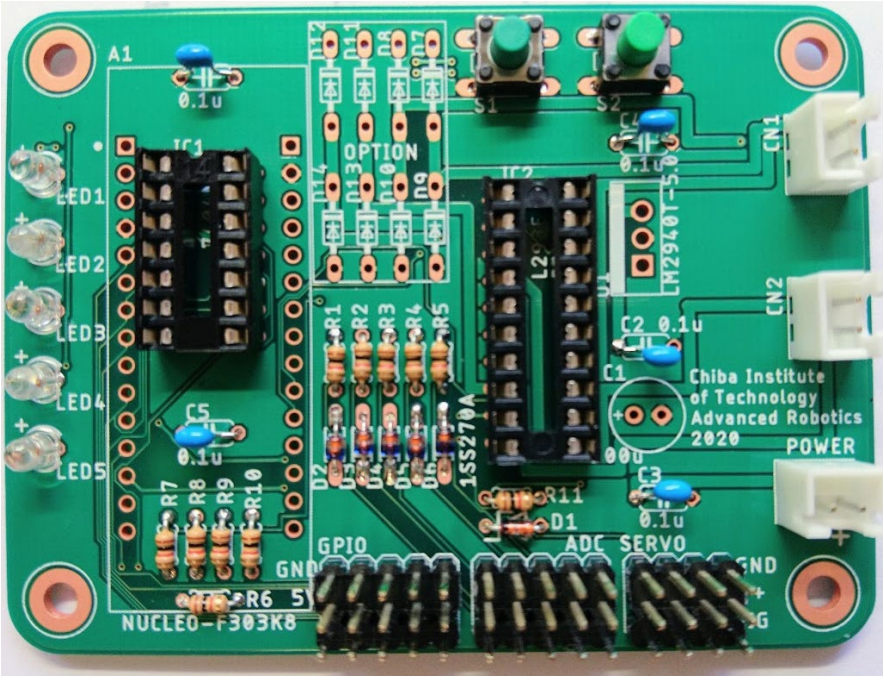


逆に取り付けても，プラグを通常とは逆に配線すると使用できます，しかし，間違いやすいのでおすすめしません．

浮かないようにするためには，ソケットのところで述べたように，１つの端子だけはんだ付けした後に，部品を押し付けながら加熱すると良いです．

９）ピンヘッダの取り付け

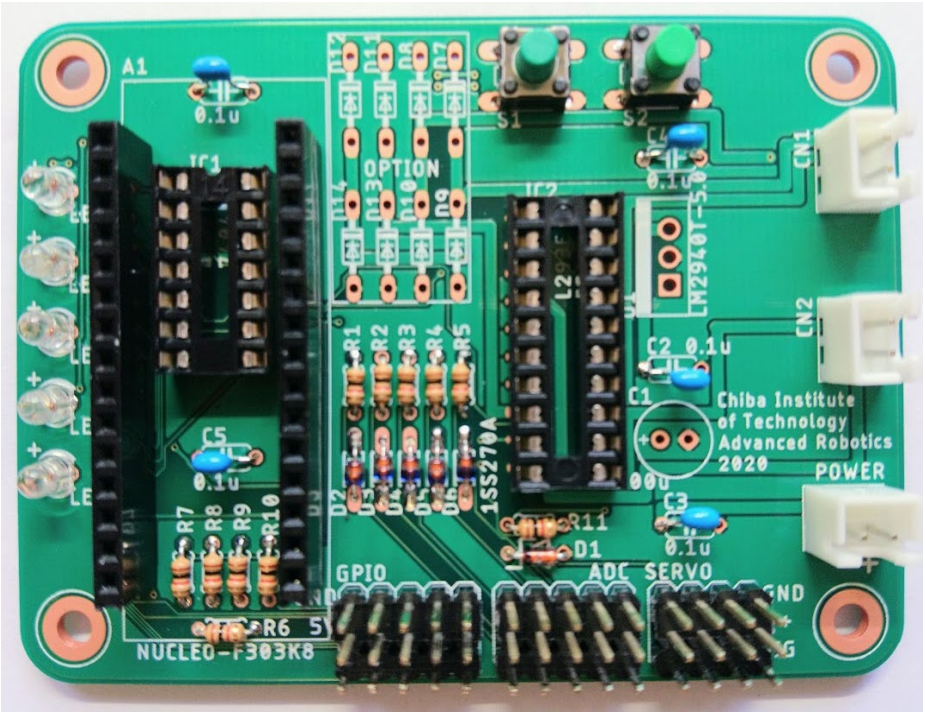
右下の剣山のようなピンヘッダを取り付けます．浮きやすいので，気をつけてください．



こちらも１端子だけはんだ付けして，押し付けながら加熱します．
やけどしやすいので，気をつけてください．

１０）ピンソケットの取り付け

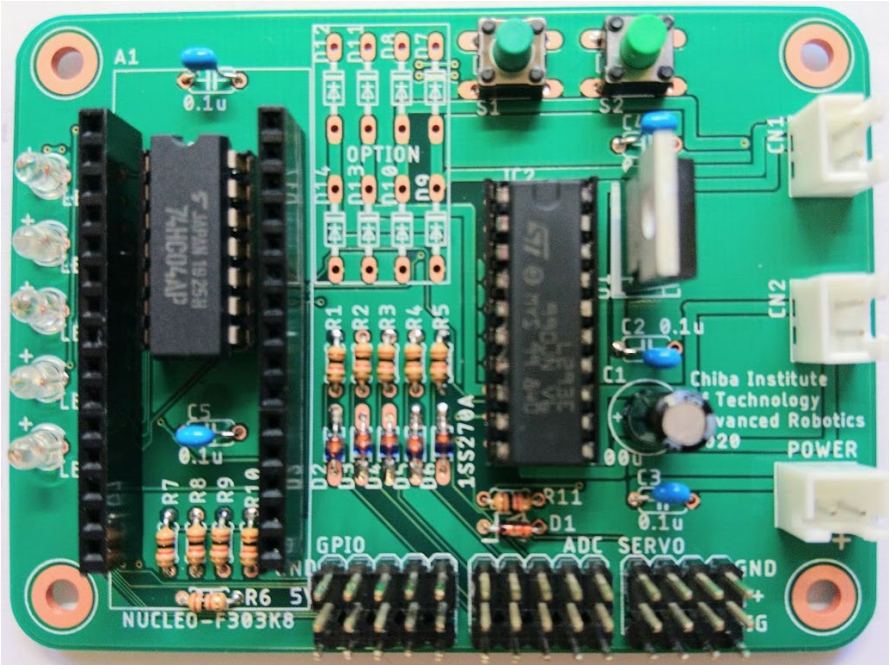
CPU(mbed)を取り付けるピンソケットを取り付けます．こちらも浮かないようにしてください．斜めに取り付けると CPU が取り付けられなくなりますので，気をつけてください．



１１）電解コンデンサ，３端子レギュレータ，IC の取り付け

最後に残った電解コンデンサと３端子レギュレータを取り付けます．電解コンデンサは**向きがあります**ので，気をつけてください．完了したら，IC をソケットに取り付けてください．向きがありますので，間違えないようにしてください．

電解コンデンサを逆に取り付けると通電時に弾けますので気をつけてください．
（電源を逆に取り付けた場合も同じく弾けます．）

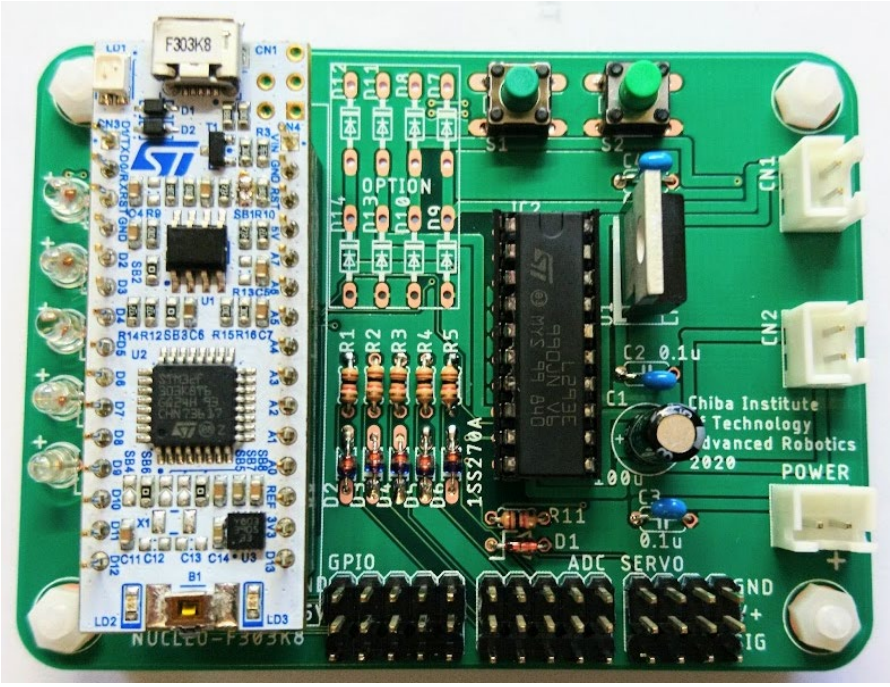


１２）CPU と六角スペーサの取り付け

CPU をソケットに取り付けて下さい．向きや場所に気をつけて取り付けて下さい．最後にショートしないように六角スペーサを取り付けましょう．

ずれた状態で取り付けていないか確認してから電源を入れてください．

ずれていると，電子回路が故障するおそれがあります．



３．プログラム環境の構築

mbed の開発環境にはオフラインとオンラインがあります．オンラインはお手軽ですが，インターネットが利用できる環境でしか使用できません．そのため，実習はオフラインコンパイラを使用して実習を行います．オフラインコンパイラを含めた環境構築の方法を示します．

なお，OS は Windows11 を想定しています．

３．１　プログラムのダウンロード

まずは開発環境構築に必要なプログラムのダウンロードを行います．以下のアドレスにアクセスしてダウンロードを行ってください．なお，以下は動作確認を行なったバージョンですが，最新のバージョンでも動作すると思います．

１）Keil uVision5 (935MB)※
　http://www.keil.com/download/product/
　MDK539A.EXE

mbed の標準的な開発環境で，無料で使用することができます．

２）Tera Term 5.2
　https://github.com/TeraTermProject/teraterm/releases
　teraterm-5.2.exe

通信用のアプリケーションです．

３）プロジェクトのテンプレート
　manaba　もしくは以下のサイト
　https://os.mbed.com/teams/adrobo/code/adrobo_template/

実習用に作成した環境とサンプルプログラムです．これを改造して独自のプログラムを作成します．

※１）のソフトウェアはダウンロードに１時間程度かかることがあります．ご注意下さい．

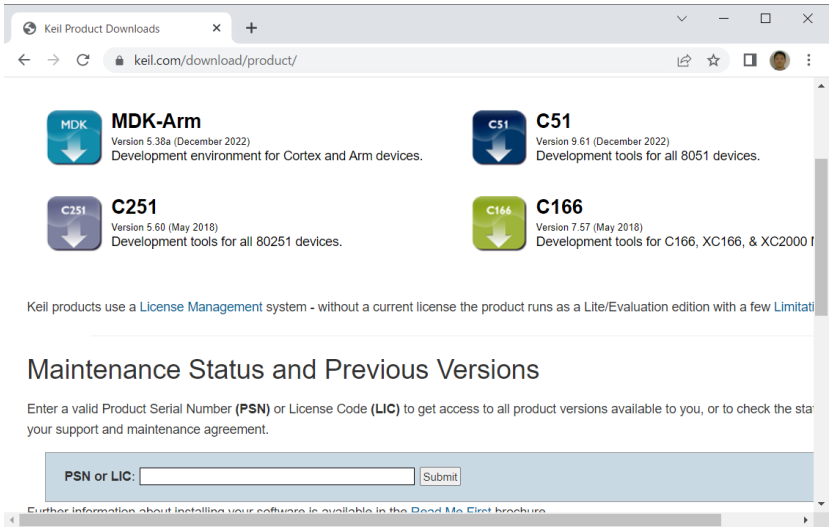
３．２　MDK-ARM のインストール

MDK-ARM は mbed の開発環境で，C++言語でプログラムを作成して，ビルドして，転送を行うことができます．

１）ウェブブラウザに URL を入力してください．

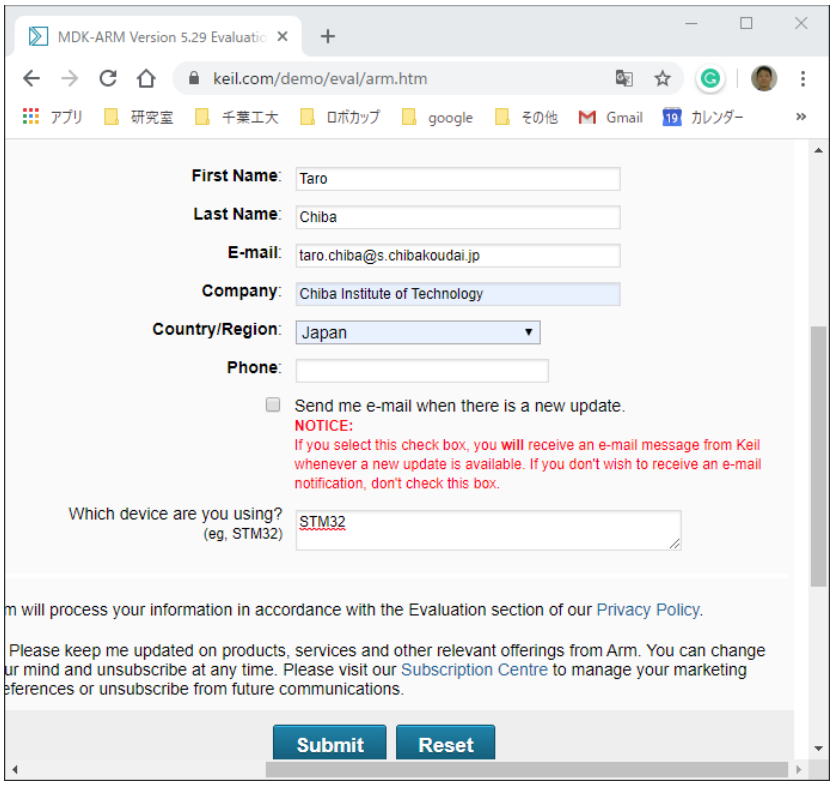
http://www.keil.com/download/product/

２）ダウンロードサイドの[MDK-ARM]をクリックしてください．



835MB ほどありますので，ダウンロードに時間がかかることがあります．

３）質問に回答してください．

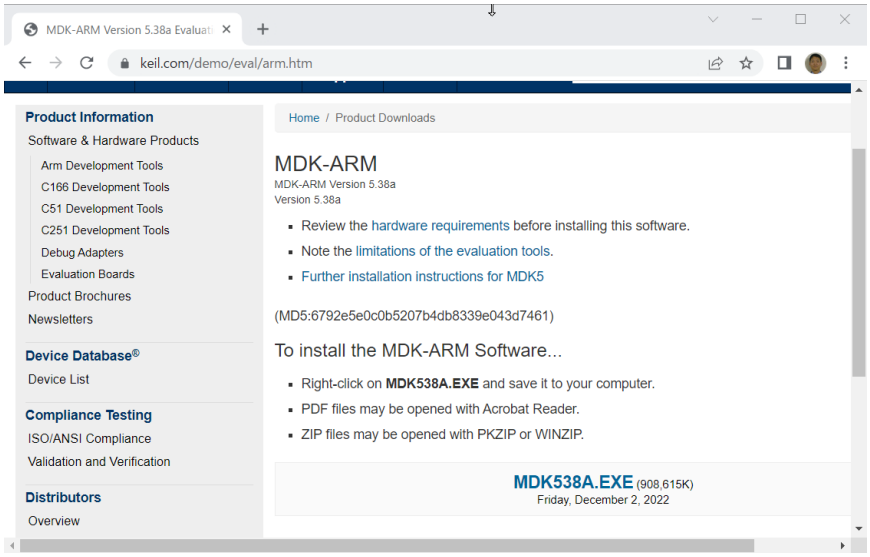


大学の情報は以下となります．

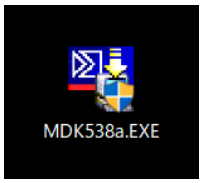
Company:
”Chiba Institute of Technology”

４）入力したら[Submit]をクリックしてください．

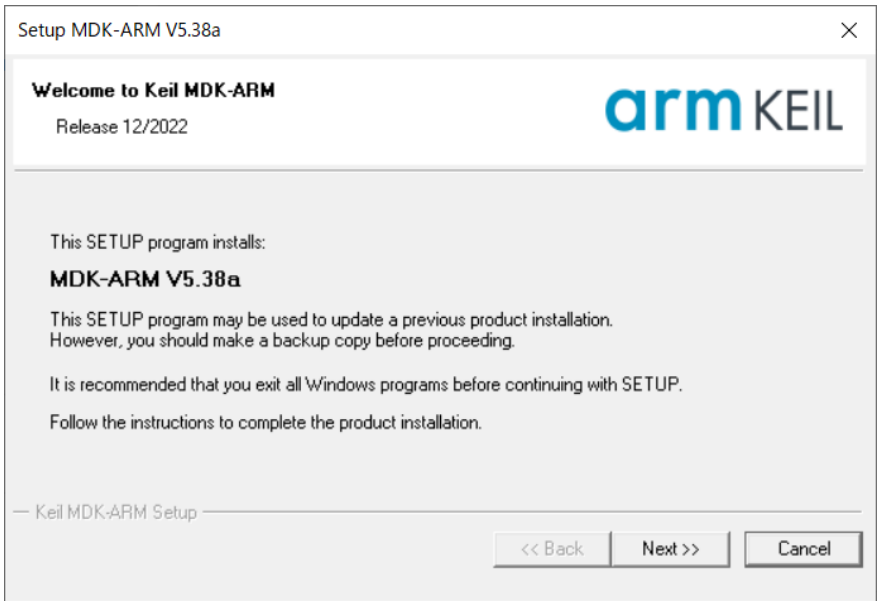
５）”MDK539A.EXE”をクリックしてダウンロードして下さい．



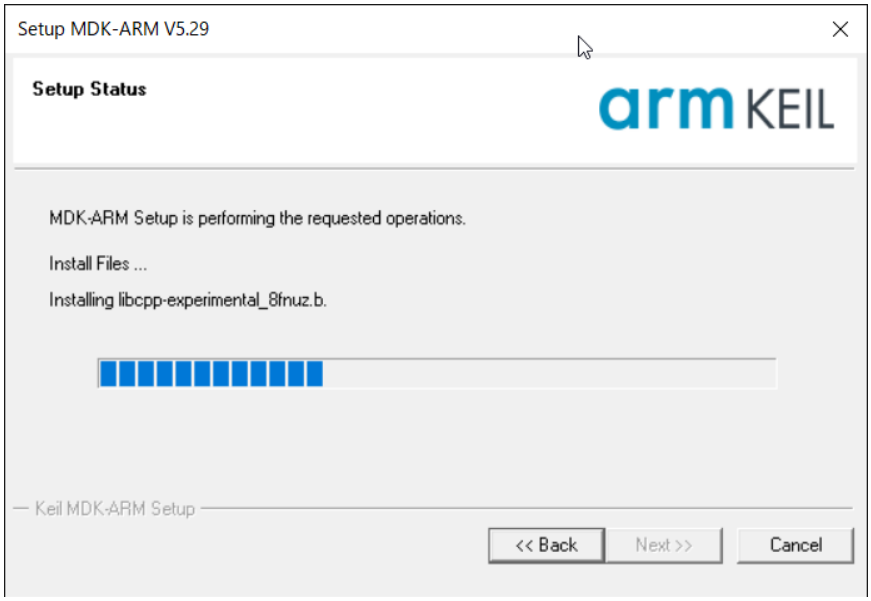
６）ダウンロードした”MDK539A.exe”をダブルクリックして下さい．



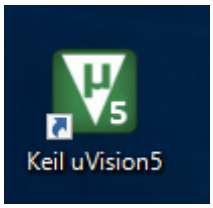
７）インストーラが起動しますので，適宜質問に回答しながら，[Next>>]を押してください．



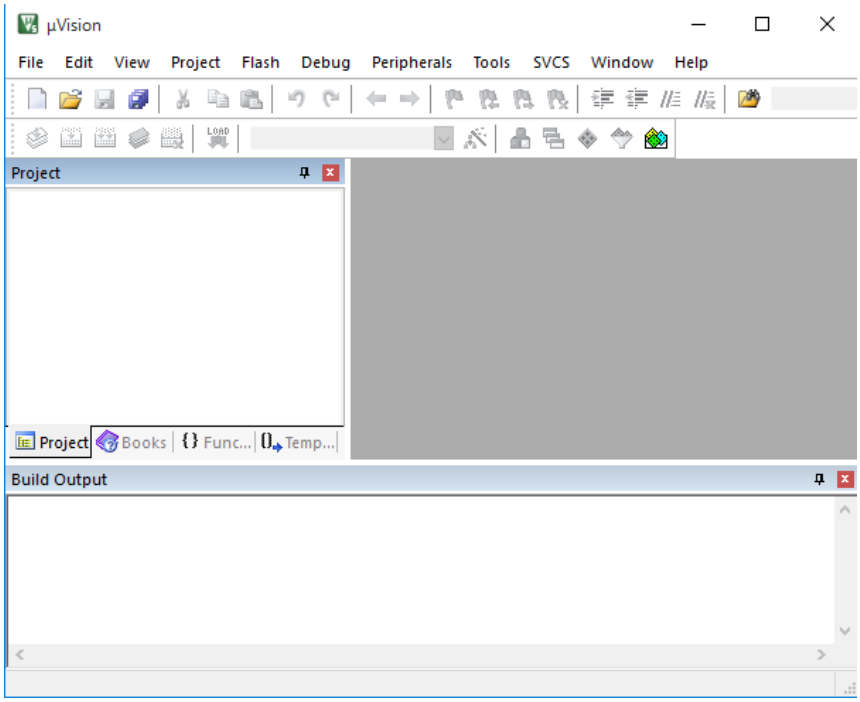
８）インストールが開始されます．



９）インストールが終了したら，”KEIL uVision5”のショートカットをダブルクリックして下さい．



１０）こちらの開発環境が起動することを確認して下さい．



３．３　Tera Term のインストール

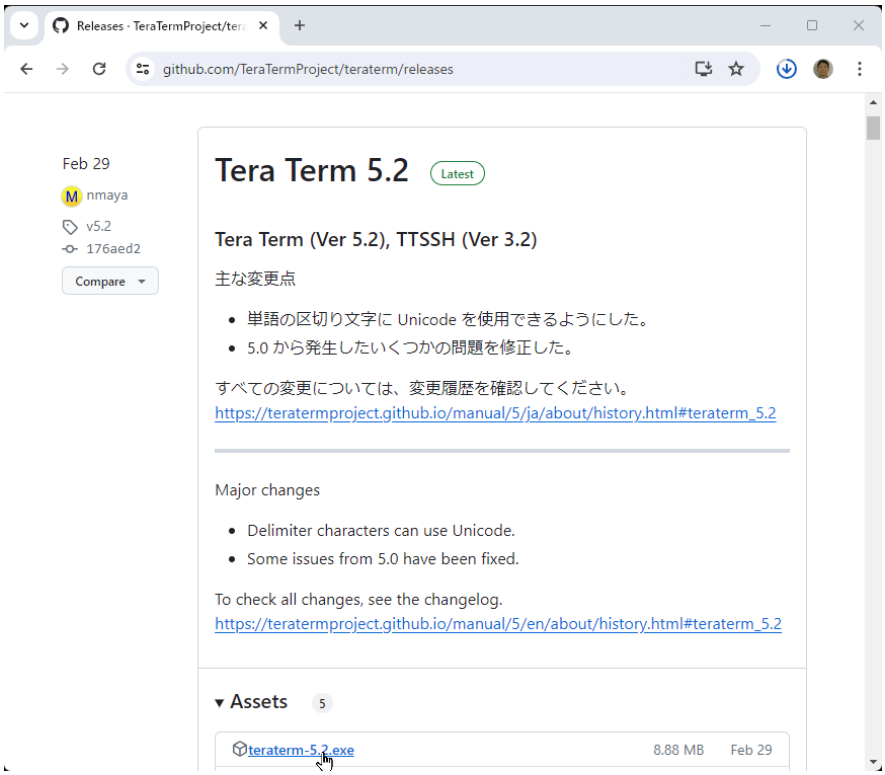
Tera Term はシリアル通信によく使用されるソフトウェアで，この実習では mbed との通信に使用します．

シリアル通信により，mbed からメッセージを表示することができます．

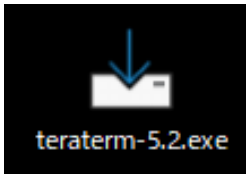
１）ウェブブラウザに URL を入力してください．
https://github.com/TeraTermProject/teraterm/releases

２）ダウンロードサイドの[teraterm-5.2.exe]をクリックして下さい．

バージョンはどれでも構いません．

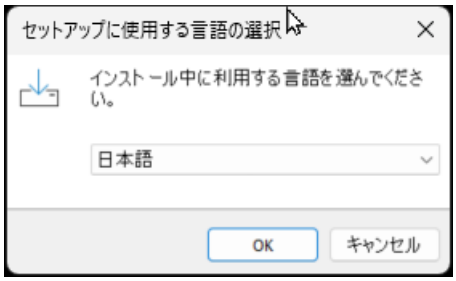


４）ダウンロードした”teraterm-4.106.exe ”をダブルクリックして下さい．



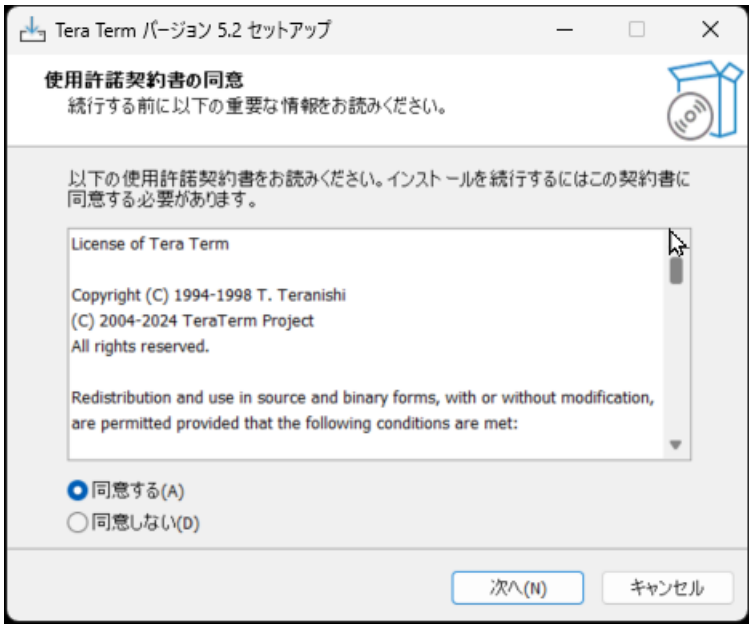
５）ユーザアカウント制御には「はい」を選択して下さい．

6）インストールする言語を選んで下さい．

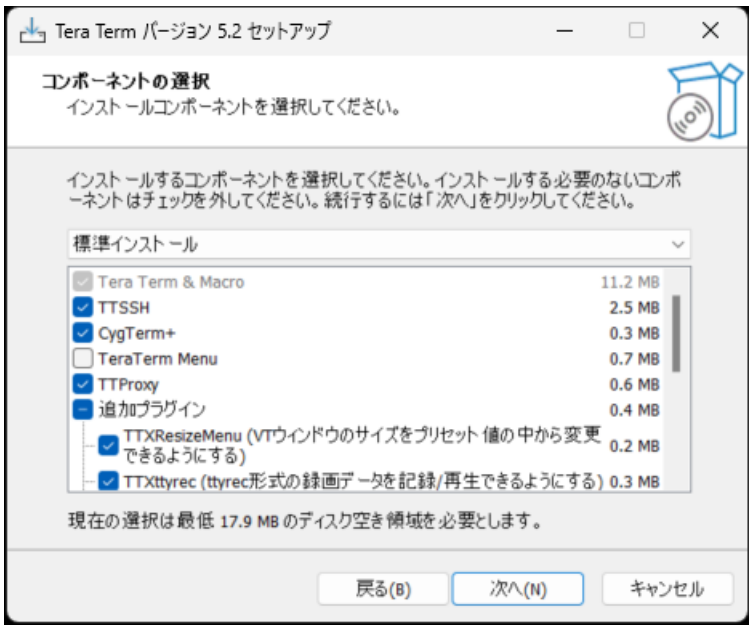


日本語以外でも構いません．

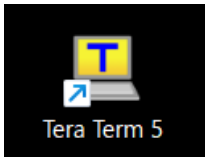
7）セットアップが起動して，同意するとインストールを開始します．



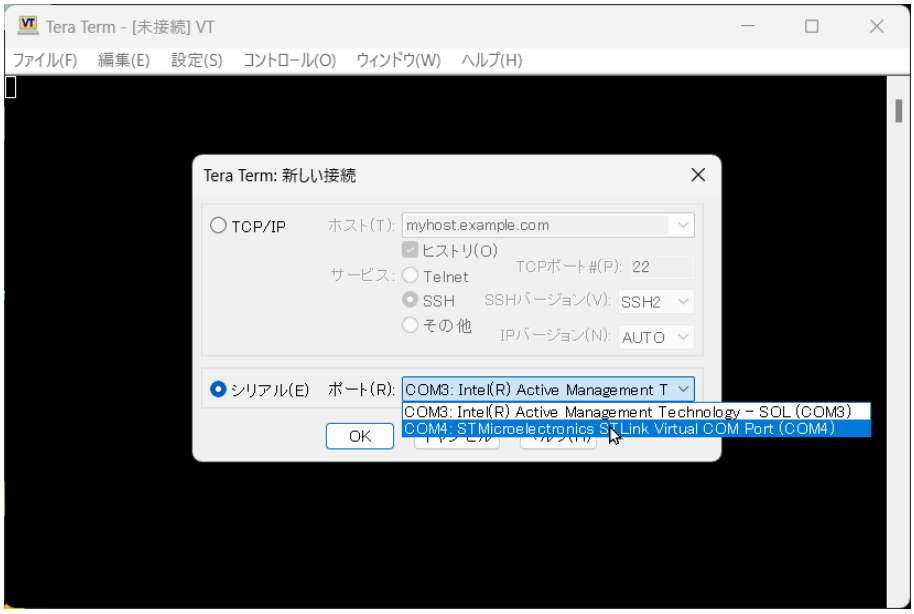
8）[標準インストール]を選択して下さい．



９）”Tera Term”のショートカットをダブルクリックして下さい．



１０）こちらの開発環境が起動することを確認して下さい．以上で，Tera Term のインストールは終了です．なお，mbed を USB で接続すると，シリアルポートを介して，mbed と通信できるようになります．
ただし，図のように COM ポートの選択などの設定が必要です．



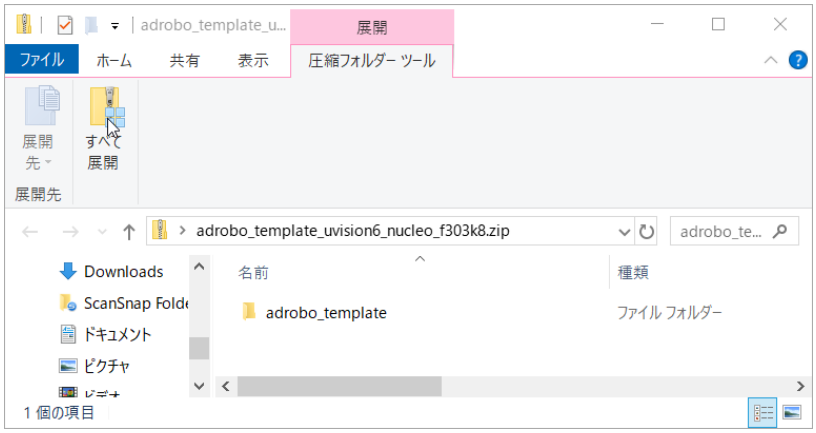
３．４　サンプルプログラムのダウンロードと動作確認

本実習で使用する制御回路は，この実習のために独自に開発したものです．また，開発しやすいように独自のライブラリも用意しています．ここでは，サンプルプログラムをダウンロードして動作を確認します．

１）manaba もしくは GitHub から以下をダウンロードしてください．
adrobo_template_uvision6_nucleo_f303k8.zip
GitHub: https://github.com/yasuohayashibara/control_board
ちなみにオンラインコンパイラで使用したい場合は，以下のアドレスに同様のファイルがアップロードされています．
https://os.mbed.com/teams/adrobo/code/adrobo_template/

２）圧縮ファイルの展開
圧縮ファイル adrobo_template_uvision6_nucleo_f303k8.zip を展開して下さい．

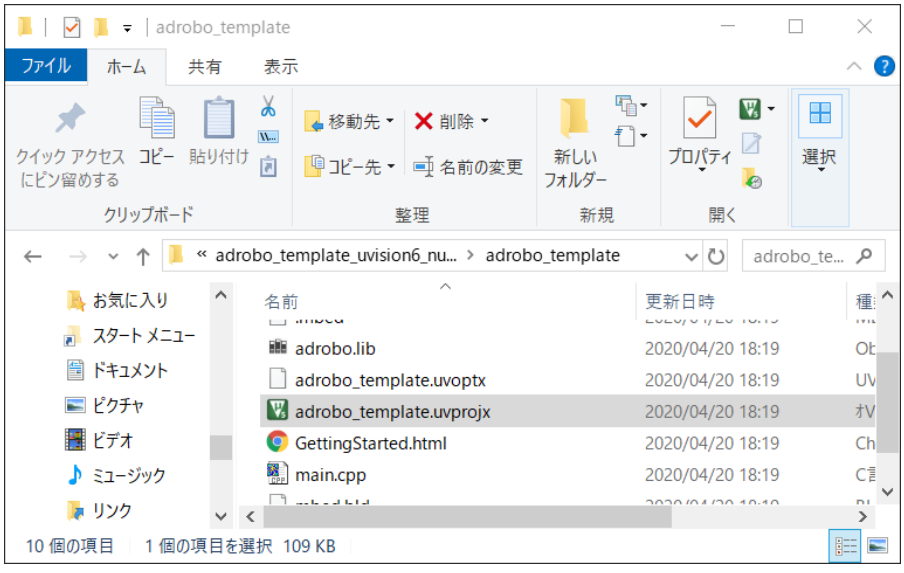
Windows の標準機能で展開できます．



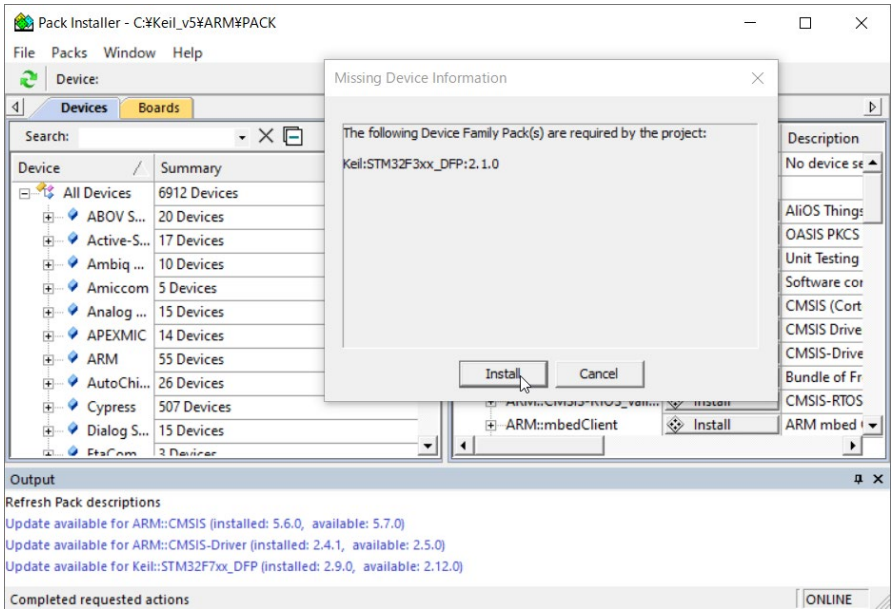
３）展開したファイルを確認します．また，以下のファイルをダブルクリックします．
[adrobo_template.uvprojx]

関連付けがうまくできていないと起動しないことがあります．

関連付けを行うと起動できるようになります．

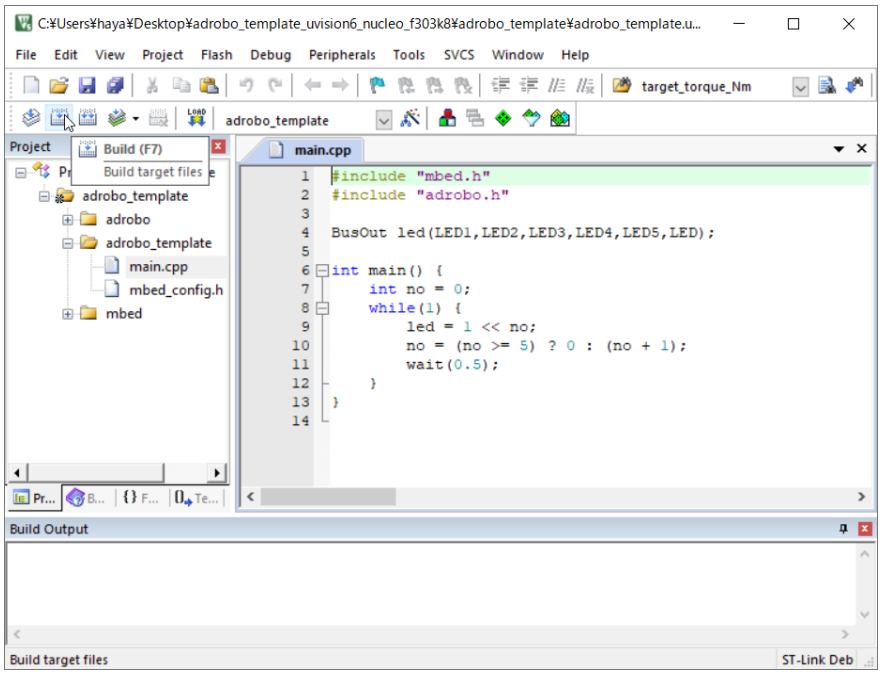


４）KEIL が起動します．
１回目は Pack Installer が起動しますので，必要な環境をインストールしてください．



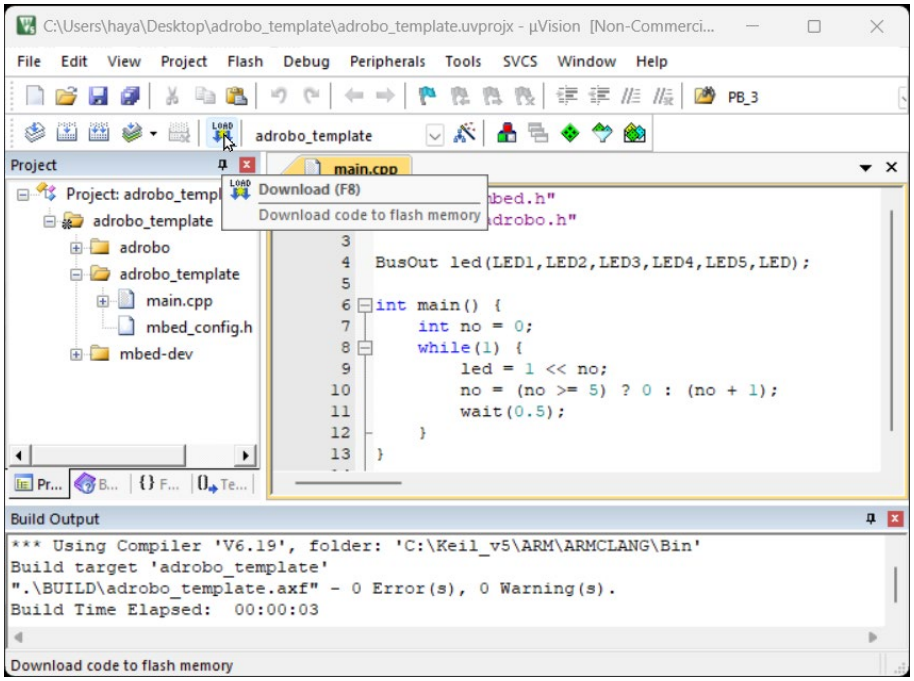
５）ソースコードを編集できる状態になります．以下の[Build]アイコンをクリックするか，[F7]を押してビルドしてください．

左の[main.cpp]をクリックして，プログラムを見てみましょう．



６)コンピュータと mbed を USB ケーブルで接続して下さい.mbed の LED ランプが点灯することを確認してください.

７）以下の[Download]アイコンをクリックして下さい.



８）mbed の LED がしばらく点滅します．点滅が終わると，制御回路上の LED が順番に点滅する様子が見られます.

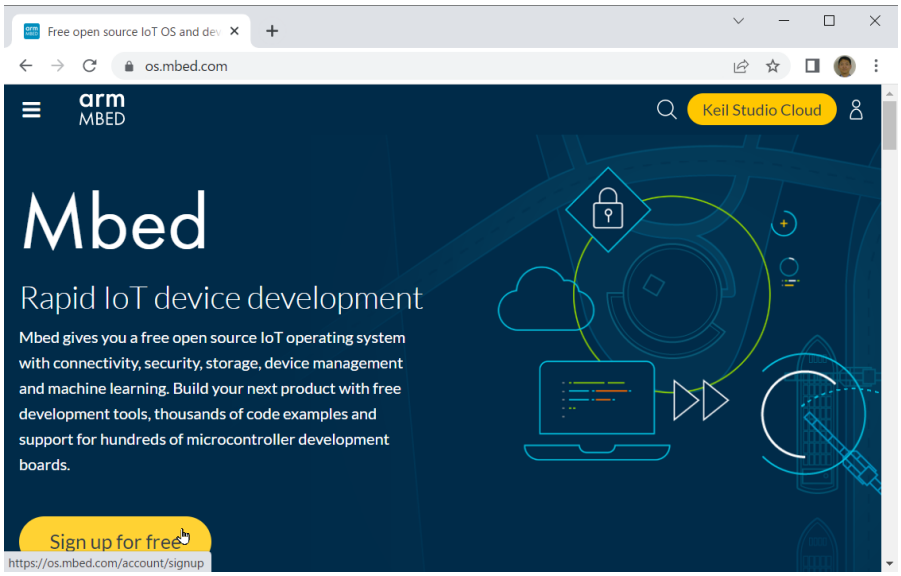
９）条件によっては書き込めないこともあります．その場合は，mbed のリセットボタンを押しながら書き込んで,リセットボタンを離して書き込むと書き込めることがあります.

リセットボタンを押しながら書き込むとエラーになりますが，その後書き込めることがあります.

３．５　オンラインコンパイラの設定

　オンラインコンパイラはロボット体験実習ではあまり使用しませんが，便利ですので，設定の方法を簡単に以下に示しておきます．

１）まずは，mbed のウェブページにアクセスして，[Sign up]をクリックしてください．
mbed のアドレス　https://os.mbed.com/

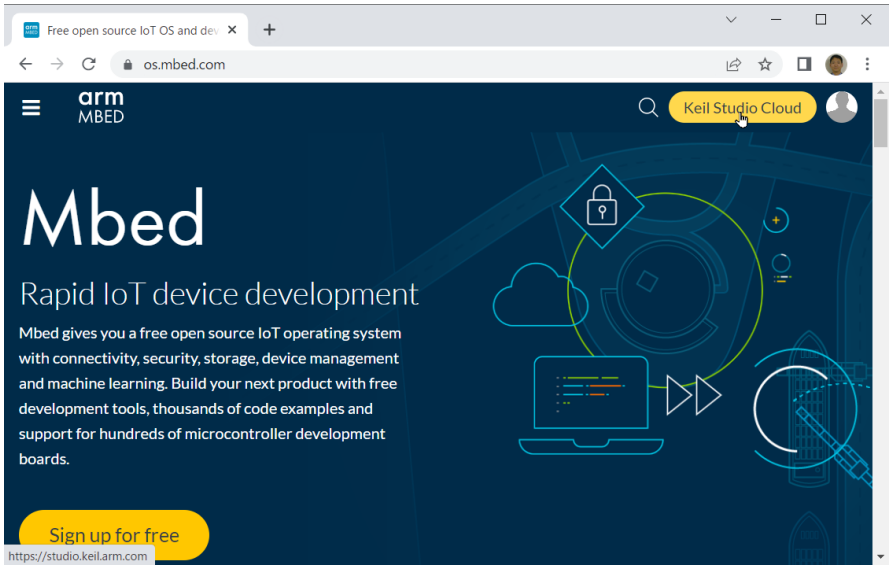


２）Log in の画面が出てきたら，[Create an account]のリンクをクリック

３）　[Sign up]をクリック

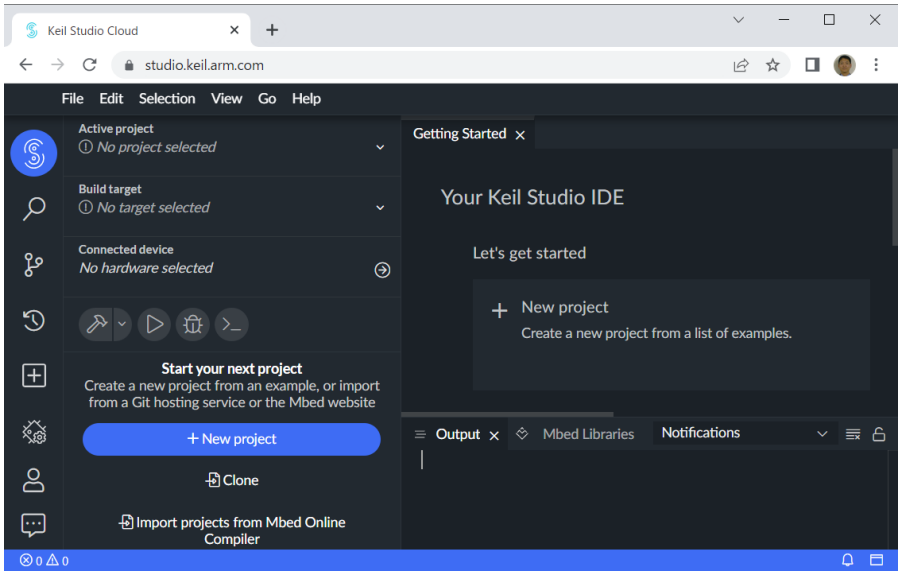
４）質問に回答して，[Sign up]をクリック

５）右上の[Keil Studio Cloud]をクリック



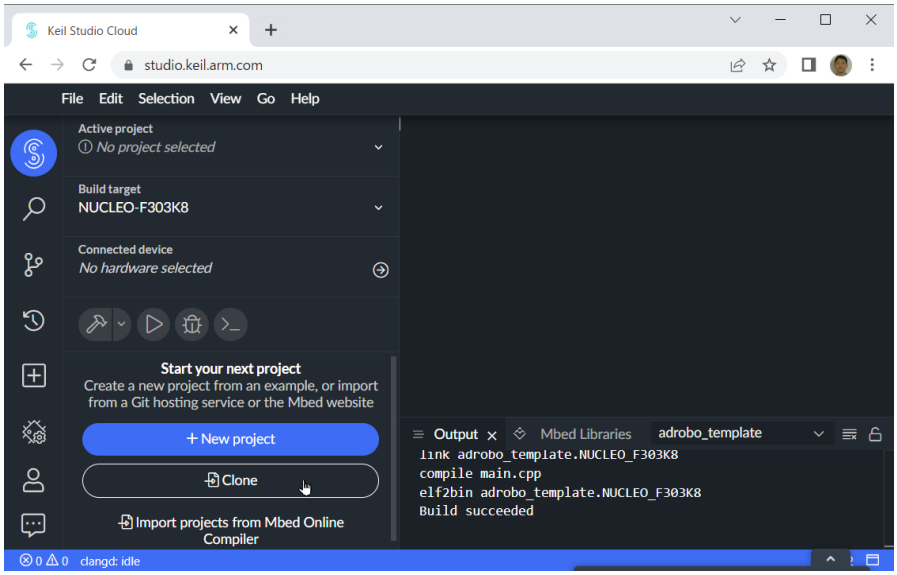
６）開発環境が起動します．

こちらを使用してソフトウェアの開発を行うこともできます．ただし，みなさんが一斉にネットワークにアクセスすると，動作が遅くなったり不安定になったりするため，原則的にはオフラインの開発環境を使用するようにしてください．



7）プログラムのインポート

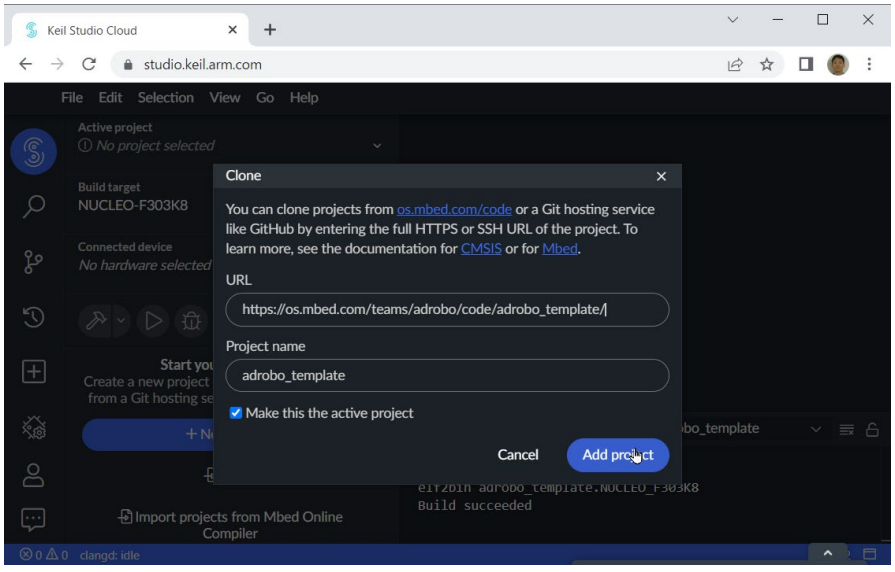
プログラムを読み込んで，コンパイルしてみましょう．



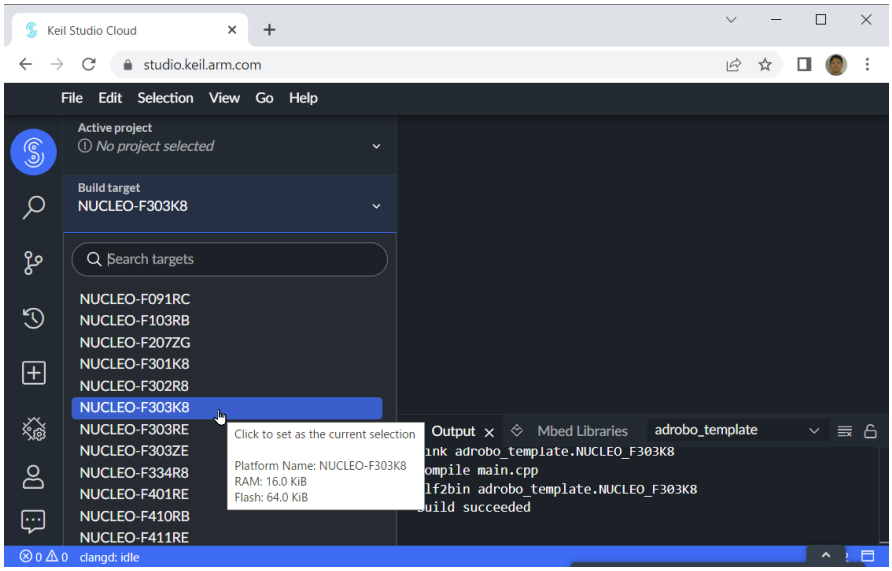
8）URL からインポートする

実習のテンプレートが以下にありますので，インポートしてください．

https://os.mbed.com/teams/adrobo/code/adrobo_template/

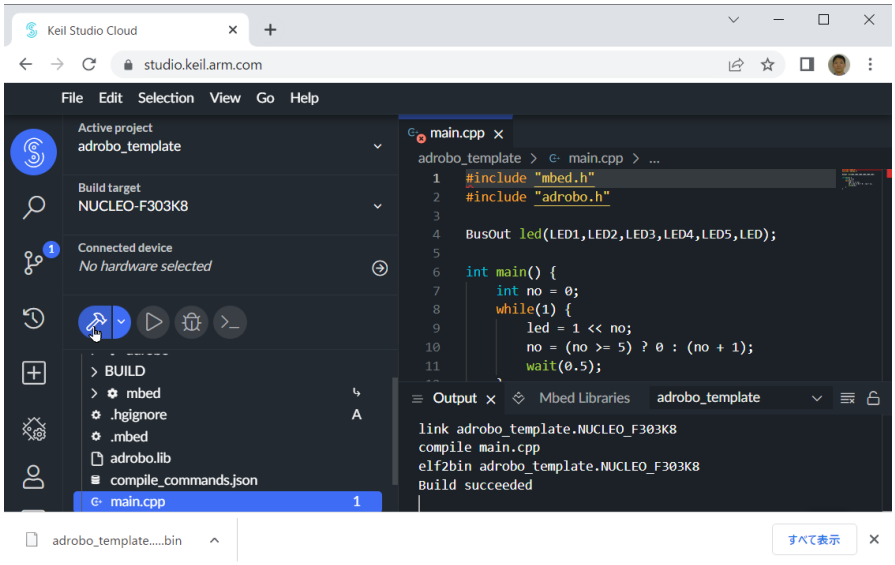


９）Build target を NUCLEO-F3030K8 にする.



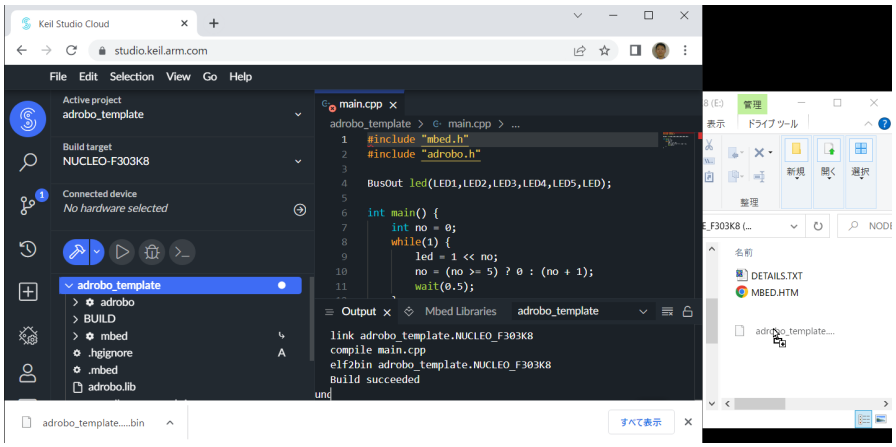
１０）コンパイルする.

コンパイルが成功すると実行ファイルがダウンロードされます.



１１）mbed に書き込み

mbed を USB ケーブルで PC に接続すると USB メモリとして認識されます．ダウンロードしたファイルを mbed のディレクトリにコピーすることで作成した実行ファイルを書き込むことができます．



３．６　FAQ

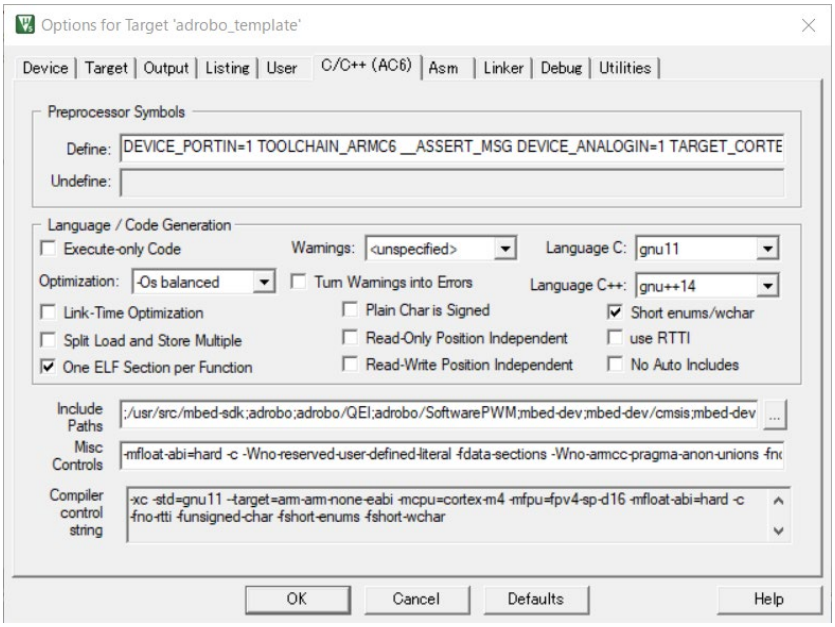
以下，報告されたよくある問題です．

１）KEIL から Nucleo に Download すると突然停止

Bluetooth を OFF にすると改善することがあります．

２）KEIL で容量が大きくて書き込みできないとのメッセージが出力

コンパイルの -Os balanced を選択する．



３）KEIL から Nucleo に Download できない

add device から STLink を選んでデバイスを接続

４．サンプルプログラム

コンピュータプログラミングは，習うより慣れた方が良いところもありますので，サンプルプログラムを入力しながら，動作を確認していくことが修得への近道となります．ここでは，基本となるサンプルプログラムを示しながら，制御回路の動作を確認していきましょう．

４．１　LED の点灯

まずは，LED の点灯のサンプルプログラムです．

```
#include ”mbed.h”

BusOut led(D13);

int main() {
  led = 1;
  wait(10);
}
```

プログラム 1-1 ワンボードコンピュータの LED 点灯

ワンボードコンピュータの LED が点灯したと思います．では，次に制御回路の LED を点灯させてみましょう．

```
#include ”mbed.h”

BusOut led( D2,D4,D5,D7,D8);

int main() {
  led = 1;
  wait(10);
}
```

プログラム 1-2 制御回路の LED 点灯（緑色）

制御回路に取り付けた５つの LED の１つが点灯したと思います．では，次に隣の LED を点灯させてみましょう．LED が２進数で ON/OFF が表されることに注意しながら，LED を自由に点灯できるようにしましょう．

４．２　時間による制御

点灯ができたら，次は時間によって点滅するプログラムを作ります．0.5 秒毎に LED の ON と OFF を繰り返し行います．

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8);

int main() {
  while(1){
    led = 1;
    wait(0.5);
    led = 0;
    wait(0.5);
  }
}
```

プログラム 2-1 LED 点滅

0.5 秒ずつ停止して，LED の ON と OFF を繰り返します．このように，時間を使った制御も簡単にできます．

少し難しい命令も含まれていますが，順番に LED を点滅させるプログラムを以下に示します．

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);

int main() {
    int no = 0;
    while(1) {
        led = 1 << no;
        no = (no >= 5) ? 0 : (no + 1);
        wait(0.5);
    }
}
```

プログラム 2-2 順番に LED を点滅

４．３　スイッチの入力

スイッチの入力により LED を点滅させてみましょう.

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);
BusIn sw(D3, D6);

int main() {
    sw.mode(PullUp);
    while(1) {
        led = sw;
    }
}
```

プログラム 3-1 スイッチによる LED の点灯・消灯

スイッチの入力の ON/OFF により LED を点灯させています.このとき,スイッチを離すと点灯して,押すと消灯することが分かると思います.

実は mbed4adrobo の基板上のスイッチは押さない時が High(1)で,押しているときが Low(0)になります．これは電子回路を学ぶときに再度解説しますが，普段の感覚とは逆になることを覚えておいて下さい.

なお，これと関連するのですが,

```
sw.mode(PullUp);
```

という行があります．スイッチを押さない時には＋電源に引っ張る(PullUp)モードを指定しています．ロボット電子回路で詳細に関しては解説しますが，これはスイッチを押していない時に，入力が確実に High になるようにするために必要な命令となります.

ビットの操作に関して確認すると，スイッチの入力と出力信号の間には以下の関係があります.

表　スイッチの状態に対するコンピュータの値（2 進数）

		SW1	
		ON	OFF
SW2	ON	00 (=0)	2 進数 01 (=1)
	OFF	10 (=2)	2 進数 11 (=3)

()内は 10 進数を表す

次に，SW1 を押すと LED1 が点灯するようにしましょう．このとき，SW1 が ON になると 0 となるため，以下のように条件分岐をさせれば良いことが分かります．SW２を使用しない宣言になっていることに気をつけてください．

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);
BusIn sw(D3);

int main() {
  sw.mode(PullUp);
  while(1){
    led = sw == 0 ? 1 : 0;
  }
}
```

プログラム 3-2 スイッチによる LED の点灯・消灯

このように，制御回路だけで学べることはたくさんあります．サンプルプログラムを基に，LED とスイッチを使用したプログラムをいくつか作ってみましょう．

【課題】

SW1 を押すと左から右へ，SW2 を押すと右から左へ LED の光が流れるようなプログラムを作ってみましょう．

４．４　USB を介したシリアル通信

メッセージを表示してみましょう．これらは，USB を通じたシリアル通信により，文字列データを送受信することで行います．

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);

int main() {
  while(1) {
    printf(”Hello World¥r¥n”);
    wait(0.5);
  }
}
```

プログラム 4-1 メッセージの送信

ちなみに，上記のプログラムでは，LED は使いませんが，初期設定を行っています．使わない時でも初期設定をしないと中途半端に LED が点灯するなど問題が発生するときがありますので，本サンプルプログラムでは常に初期設定を行うようにします．

メッセージは Teraterm などで見ることができます．設定が間違っていていると，見られなかったり間違った文字が出力されることなどがあります．

それでは，これを応用して，スイッチの状態を表示するようにしてみましょう．

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);
BusIn sw(D3, D6);

int main() {
  int state = -1;
  sw.mode(PullUp);
  while(1) {
    if (state != sw) {
      if (sw & 0x01) printf(”SW1 OFF ”);
      else printf(”SW1 ON ”);
      if (sw & 0x02) printf(”SW2 OFF¥r¥n”);
      else printf(”SW2 ON¥r¥n”);
      state = sw;
    }
  }
}
```

プログラム 4-2 スイッチの状態の表示

さらに，コンソールから入力したメッセージをループバックさせてみましょう．（文字を入力して Enter を押すと表示されます．）

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);

int main() {
  char buf[100];
  while(1) {
    scanf(”%s”, buf);
    printf(”Return Message: %s¥r¥n”, buf);
  }
}
```

プログラム 4-3 受信したメッセージの送信（ループバック）

実行しても入力した文字が表示されるだけと思うかもしれませんが，一度 mbed に送られてから，それがエコーバックされて表示されています．

さらに,コンソールから入力した数字でLEDを点滅させてみましょう.

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);

int main() {
  char buf[100];
  while(1) {
    scanf(”%s”, buf);
    printf(”LED ON: %s¥r¥n”, buf);
    int n = atoi(buf);
    if (n > 0) led = led ^ (1 << (n - 1));
  }
}
```

プログラム 4-4 受信したデータによるLEDの点灯・消灯

1から5までの数字を入れると，それに対応するLEDが点灯，消灯します．このように，USB を介して遠隔操作を行うこともできます．この遠隔操作の方法は，そのままロボットの遠隔操作にも利用することができます.

４．５　モータの制御

モータの制御を行います．CN1 の端子にモータを接続して下さい．なお，本サンプルプログラムから，確認のためにモータなどの外付けの部品が必要となります．

```
#include ”mbed.h”
#include ”Motor.h”

BusOut led(D2,D4,D5,D7,D8,D13);
Motor motor1(A1, A2);

int main() {
  float delta = 0.1f, i = 0.0f;
  while(1) {
    i += delta;
    if (fabs(i) >= 1.0f) delta *= -1.0f;
    printf(”%f¥r¥n”, i);
    motor1 = i;
    wait(0.5);
  }
}
```

プログラム 5-1 モータの制御

なお，Motor クラスでは，最大出力(motor = 1 の時)でも電源の 25%しか出力しない設定になっています．これを変更するために，

```
setMaxRatio(float max_ratio)
ratio: デューティ比の最大値(0.0 から 1.0，初期設定 0.5)
```

というメソッドがありますが，問題ない場合はそのまま使用してください．

なお，２個めのモータを使用する場合は，
Motor motor2(D11, D12);
と定義して下さい．
（変数名 motor2 は自由に設定して頂いて構いません）

４．６　サーボモータの制御

SERVO にサーボモータを接続して動作を確認します．サーボモータの初期位置は個々に異なりますので，適正な値に調整して下さい．

```
#include ”mbed.h”
#include ”SoftwarePWM.h”

BusOut led(D2,D4,D5,D7,D8,D13);
PwmOut servo0(D9), servo1(D1), servo2(D0);
SoftwarePWM servo3(D10);

int main() {
  servo0.period_ms(20);
  servo1.period_ms(20);
  servo2.period_ms(20);
  servo3.Enable(0,20000);
  while(1) {
    for(int i = 400; i <= 2000; i += 100){
      servo0.pulsewidth_us(i);
      servo1.pulsewidth_us(i);
      servo2.pulsewidth_us(i);
      servo3.SetPosition(i);
      wait(0.5);
    }
  }
}
```

プログラム 6-1 サーボモータの制御

次に初期位置の角度をボタンを使って調べてみましょう.

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);
BusIn sw(D3, D6);
PwmOut servo1(D9);

int main() {
  int val = 1500, val0 = 0;
  servo1.period_ms(20);
  sw.mode(PullUp);
  while(1) {
    if (!(sw & 1)) val += 10;
    if (!(sw & 2)) val -= 10;
    if (val != val0){
      servo1.pulsewidth_us(val);
      printf(”%d[us]¥r¥n”, val);
      val0 = val;
    }
    wait(0.02);
  }
}
```

プログラム 6-2 サーボモータの初期位置の調査

４．７　A/D 変換によるセンサ値の取得

アナログ入力の値を取得しましょう．PSD 距離センサなどを接続して，値の変化を確認して下さい．

```
#include ”mbed.h”

BusOut led(D2,D4,D5,D7,D8,D13);
AnalogIn ad[] = {A0};

int main() {
  while(1) {
    printf(”%f¥r¥n”, ad[0].read());
    wait(0.2);
  }
}
```

プログラム 7-1 A/D 変換の結果の表示

PSD 距離センサとモータ制御の組み合わせで，ある一定距離を保つロボットになります．以下にサンプルプログラムを示します．

フィードバック制御と呼ばれる制御系になります．

val の前の数字を変更することで距離を変更できます．

```
#include ”mbed.h”
#include ”Motor.h”

BusOut led(D2,D4,D5,D7,D8,D13);
AnalogIn ad[] = {A0};
Motor motor1(A1, A2);
Motor motor2(D11, D12);

int main() {
  while(1) {
    float val = ad[0].read();
    printf(”%f¥r¥n”, val);
    float i = (0.6 - val) * 3.0f;
    motor1 = i;
    motor2 = -i;
    wait(0.5);
  }
}
```

プログラム 7-2 PSD 距離センサを用いて一定距離を保つロボット

なお，台車は対向２輪でモータは正方向で逆回転するとしています．ロボットの形状やモータの回転の向きにより，プログラムを修正することが必要となります．

４．８　GPIO を使用した入出力

GPIO を使用して音楽を演奏してみましょう.GPIO の 1 番目の端子(D3)に圧電スピーカを接続して，以下のサンプルプログラムを実行しましょう．スイッチ(SW1)を押すと LED が点滅しながら，音が流れます.

GPIO とは，General Purpose Input / Output の略で，汎用入出力と呼ばれます

```
#include ”mbed.h”
#include ”Sound.h”

BusOut led(D4,D5,D7,D8,D13);
BusIn sw(D3);
Sound sound(D2);

int sound_data[] = {M_C4, M_D4, M_E4, M_F4, M_G4, M_A4, M_B4,
M_C5};

int main() {
  sw.mode(PullUp);
  while(1){
    led = 0;
    while(sw);
    for(int i = 0; i < (sizeof(sound_data)/sizeof(int)); i ++){
      led = 1 << (i & 3);
      sound = sound_data[i];
      wait(0.4);
      sound = 0;
      wait(0.1);
    }
  }
}
```

プログラム 8-1 GPIO を使用した音楽の演奏

次に PSD と連動させて,距離に応じて音と LED が変化する装置のサンプルプログラムを示します.

```
#include ”mbed.h”
#include ”Sound.h”

BusOut led(D4,D5,D7,D8,D13);
AnalogIn ad[] = {A0};
Sound sound(D2);

int main() {
  while(1) {
    float val = ad[0].read();
    printf(”%f¥r¥n”, val);
    led = 0x0f >> (int)((0.9f - val) * 4);
    sound = 880 * val;
    wait(0.2);
  }
}
```

プログラム 8-2 距離に応じて周波数が変化

４．９　ロータリエンコーダのカウント

ロータリエンコーダ（回転角度センサ）を読み取りましょう．GPIO の１番と２番の端子にロータリエンコーダの出力を接続して，以下のサンプルプログラムを実行しましょう．モータを回転させるとロータリエンコーダの A 相，B 相の出力信号が LED の点滅で確認できます．また，ターミナルに回転時に計測したパルス数が出力されます．

```
#include ”mbed.h”
#include ”QEI.h”

BusIn in(D2, D4, D5, D7);
QEI qei1(D2, D4, NC, 48, QEI::X4_ENCODING);
QEI qei2(D5, D7, NC, 48, QEI::X4_ENCODING);

int main() {
  in.mode(PullUp);
  while(1) {
    printf(”%d, %d¥r¥n”, qei1.getPulses(),  qei2.getPulses());
    wait(0.5);
  }
}
```

プログラム 9-1 ロータリエンコーダのカウント

ここで，ビルド時にワーニングが出ることがあります．実行には問題が無いのでそのままでも構いませんが，変更する場合はワーニングが表示されているところを，以下のように変更して下さい．

```
channelA_.rise(this, &QEI::encode);
↓
channelA_.rise(callback(this, &QEI::encode));
```

４．１０　動作チェック用プログラム

```
#include ”mbed.h”
#include ”Motor.h”

BusOut led(D8,D13);
BusIn sw(D3,D6);
BusIn in(D2, D4, D5, D7);
Motor motor1(A1, A2);
Motor motor2(D11, D12);

int main() {
  in.mode(PullUp);
  sw.mode(PullUp);
  while(1) {
    motor1 = !sw[0];
    motor2 = !sw[1];
    wait(0.001);
  }
}
```

プログラム 10-1 モータが動作するかをチェックするプログラム

４．１１　速度制御

モータの PWM 制御とロータリエンコーダを組み合わせて速度の制御を行いましょう．PI 制御を使用して，モータの速度を一定に保ちます．

```
#include ”mbed.h”
#include ”adrobo.h”
#include ”Motor.h”
#include ”QEI.h”

BusOut led(LED);
Ticker control;
Motor motor_left(MOTOR11, MOTOR12);
Motor motor_right(MOTOR21, MOTOR22);
BusIn in(GPIO1, GPIO2, GPIO3, GPIO4, GPIO5);
QEI qei_left(GPIO1, GPIO2, NC, 48, QEI::X4_ENCODING);
QEI qei_right(GPIO3, GPIO4, NC, 48, QEI::X4_ENCODING);

double speed_left_ref = 0.2, speed_right_ref = 0.2;
double speed_left_lpf = 0.0, speed_right_lpf = 0.0;
const double sampling_time = 0.020;
const double move_per_pulse = 0.0005;
double i_left = 0.0, i_right = 0.0;
const double ki = 10.0, kp = 1.0;

double low_pass_filter(double val, double pre_val, double gamma) {
  return gamma * pre_val + (1.0 - gamma) * val;
}

void control_handler() {
  int enc_left = qei_left.getPulses();
  int enc_right = qei_right.getPulses();
  qei_left.reset();
  qei_right.reset();
  double speed_left = -move_per_pulse * enc_left / sampling_time;
  double speed_right = move_per_pulse * enc_right / sampling_time;
  speed_left_lpf = low_pass_filter(speed_left, speed_left_lpf, 0.4);
  speed_right_lpf = low_pass_filter(speed_right, speed_right_lpf, 0.4);
  double delta_speed_left = speed_left_ref - speed_left_lpf;
  double delta_speed_right = speed_right_ref - speed_right_lpf;
  i_left += delta_speed_left * sampling_time;
  i_right += delta_speed_right * sampling_time;
  motor_left = kp * delta_speed_left + ki * i_left;
  motor_right = kp * delta_speed_right + ki * i_right;
}

int main() {
  in.mode(PullUp);
  motor_left.setMaxRatio(1.0);
  motor_right.setMaxRatio(1.0);
  control.attach(&control_handler, sampling_time);
  led = 1;
  while(1){
    led = led ^ 1;
    wait(0.5);
  }
}
```

プログラム 11-1 速度制御

モータの回転方向とロータリエンコーダの値が増加する方向が逆の場合は最大速度で回転します．

まずは，向きが一致しているかを確認してから，プログラムを実行するようにしてください．

５．最後に

コンピュータ制御の基本を学んでいただきましたが，いかがでしたでしょうか．案外簡単に電子機器を操作できる，と感じた人が多いのではないでしょうか．現在，コンピュータ制御の世界はどんどん進化しており，使いやすいツールが世の中にどんどん生み出されています．その中でも mbed は注目の開発環境で，情報やサンプルプログラムが多く存在します．自発的に学ぶことが比較的容易な環境だと思いますので，是非とも授業だけでなく，自分でも色々と挑戦していただければと思います．