IoT 技術教材

個室の利用状況の可視化システムの構築

実習テキスト

秋田職業能力開発短期大学校

学習内容

IoT 技術を利用し個室の利用状況を可視化するシステムを構築する。本教材は、実際に動作するシステムを構築しながら IoT 技術を体験することを目的としている。

学習対象者

プログラムの基本構文を理解している者。

IT 技術に関する基本的な知識を有している者。

目次

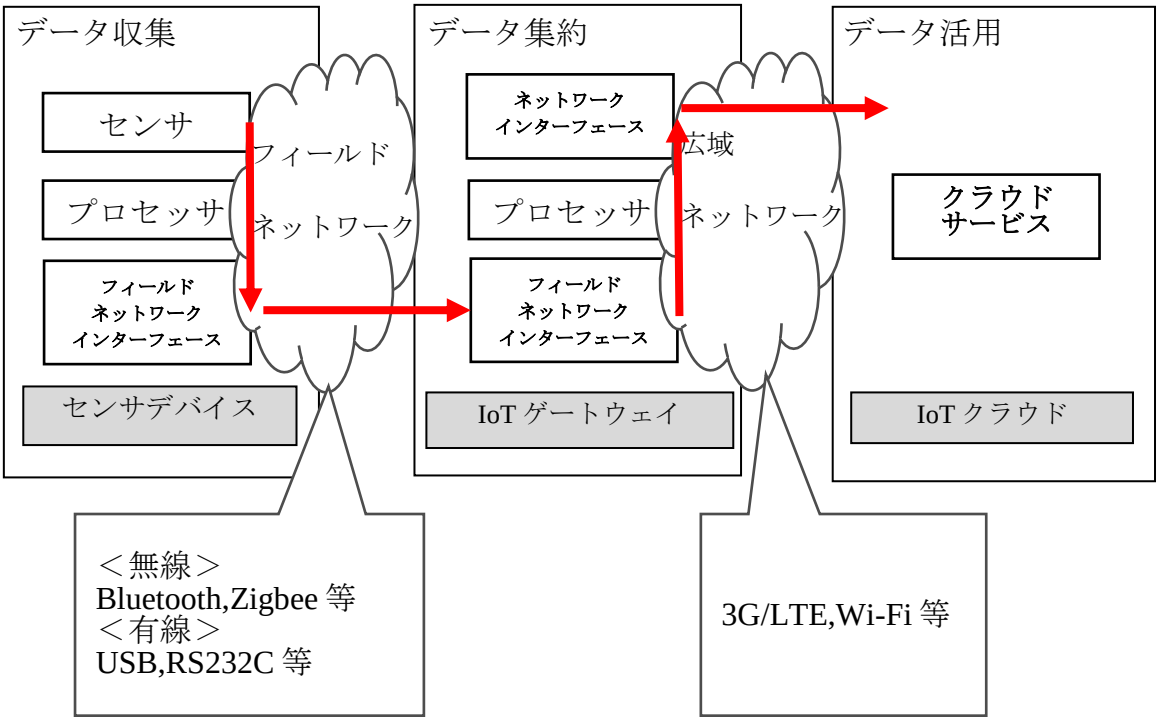
はじめに 1
IoT システムの構成 2
個室利用状況可視化システムの構成 3
センサデバイス層の構成 6
Arduino について 6
Arduino の製品ラインナップ 7
Arduino Mega の基板仕様 7
Arduino の統合開発環境 10
プログラムの書き込み 16
シリアル通信のプログラム ... 19
Bluetooth モジュール 22
センサデバイス層の実装 23
IoT ゲートウェイ層の構成 24
Windows10IoTCore 25
アプリケーションの作成 26
イベント駆動型プログラム ... 28
メソッドとプロパティ 29
構造化例外処理 33
Bluetooth 通信のプログラム ... 34
Backend as a service(BaaS)の利用 38
Firebase について 39
Firebase Realtime Database へのデータ保存 42
IoT ゲートウェイ層の実装 49
IoT クラウド層の構成 51
Firebase Realtime Database のルール設定 52
Firebase Hosting の利用 53
さいごに 58
参考文献 59

はじめに

あらゆるモノがインターネットに接続される Internet of Things(IoT)の技術は、センサやデバイスの安価化や無線技術の発展等により急速に拡大している。IoT を利用したシステムは、その仕組みや利用方法は様々である。モノにセンサを付けて情報を取得するシステムもあればネットワークを通じてモノを制御するシステムもある。また、モノにセンサを付けて集めた大量のデータ(ビックデータ)を AI（人工知能）で分析し新たな価値を生み出すなどの仕組みも見られるようになってきている。

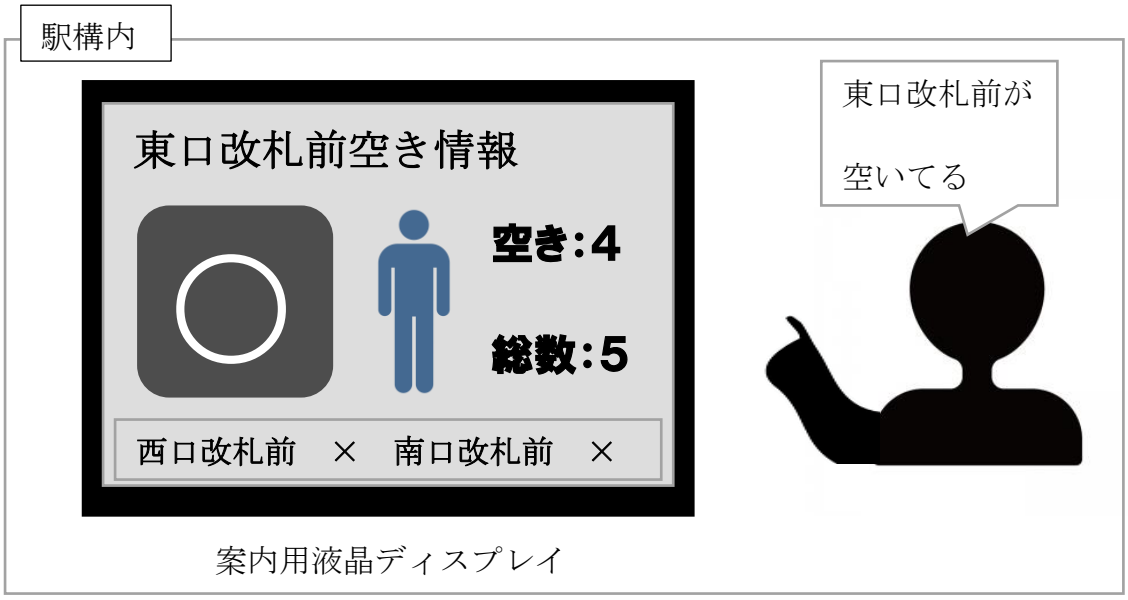
IoT システムの構成

IoT システムは、複数の技術を組み合わせたマルチレイヤーのアーキテクチャをとる。基本的なアーキテクチャは、システムの仕組みにより様々である。本教材では、システムのアーキテクチャをデータを収集するセンサデバイス層、データを集約する IoT ゲートウェイ層、データを活用する IoT クラウド層として説明する。センサデバイス層は、実際にデータを収集し場合によってはマイコン等のプロセッサで簡単な処理を行う。取得したデータを無線通信や場合によっては有線通信などを用いて IoT ゲートウェイ層へ送信する。IoT ゲートウェイ層は、送られてきたデータから必要なデータだけを取り出したりする処理等を行い IoT クラウド層へのデータ送信を行う。IoT クラウド層は、IoT ゲートウェイ経由で送られてきたデータを活用できるようにアプリケーションの実行やデータの解析などを行う。それらはクラウドサービスを活用し実現される。しかしこれは一例であり IoT システムは、システムによって利用方法や仕組みが異なりそれに合わせてアーキテクチャも変化する。また今日では、IoT に特化した通信技術、クラウドサービスも普及し始めている。[1][2]



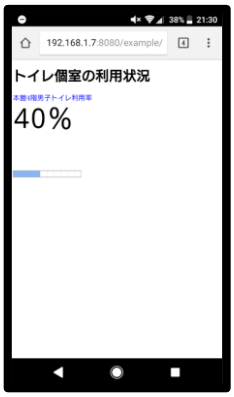
個室利用状況可視化システムの構成

実習で作成する個室利用状況可視化システムは、トイレの個室やコインロッカー、会議室等の部屋を効率よく利用してもらうために個室の利用状況をリアルタイムに通知するシステムである。実習では、トイレ個室の利用状況を可視化するシステムを題材にして進めることとする。このようなトイレ個室の利用状況可視化システムは首都圏の駅のトイレに既に導入されている。例えば、小田急電鉄株式会社は KDDI 株式会社が開発したシステムを導入し駅構内の案内用液晶ディスプレイに個室の利用状況をリアルタイムに表示するとともに専用のアプリを利用することでスマートフォンから個室の利用状況を確認できるサービスを提供している。混雑するトイレを効率よく利用してもらうためのサービスである。[3]



実習で作成するシステムの構成を以下に示す。データを収集するセンサデバイス層では、個室のドアに開閉を検知するために磁気リードスイッチを取り付けマイコン基板 Arduino を利用し各個室の利用状況を取得する。この情報を Bluetooth 通信を利用し IoT ゲートウェイ層のマイコン基板 RaspberryPi3 へ送信する。データを集約する IoT ゲートウェイ層では、センサデバイス層から受け取った個室の利用状況を液晶ディスプレイを通じて通知する処理と広域ネットワーク経由でクラウドに利用状況を送信する処理を行う。IoT クラウド層では、クラウドサービスを利用し利用者に対して個室の利用状況をリアルタイムに通知する専用の Web サイトを提供する。

データ活用：IoT クラウド層



スマートフォン
で利用状況を確認

クラウドサービス

Google 社の Firebase クラウド
・利用状況の格納
・専用 Web サイトの提供

広域ネットワーク

送信データ
個室の利用状況
送信方法
Wi-Fi または
Ethernet

データ集約：IoT ゲートウェイ層

案内用液晶ディスプレイ

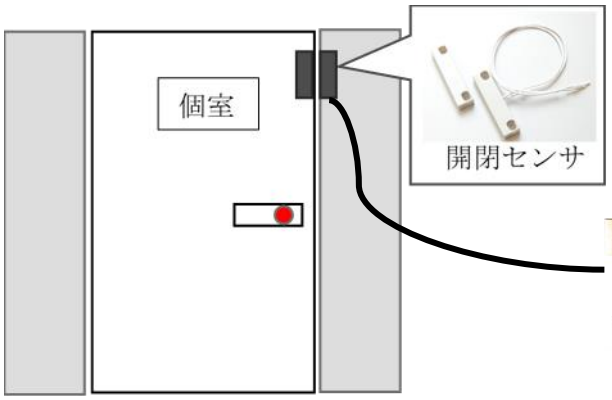
RaspberryPi3
オンボード上の通信モジュールを利用
個室利用状況の受信
・Bluetooth
クラウドへの利用状況送信
・Ethernet
・Wi-Fi

OS は Windows10IoTCore を搭載
C#でアプリケーション作成

フィールドネットワーク

送信データ
個室の利用状況
送信方法
Bluetooth 通信

データ収集：センサデバイス層

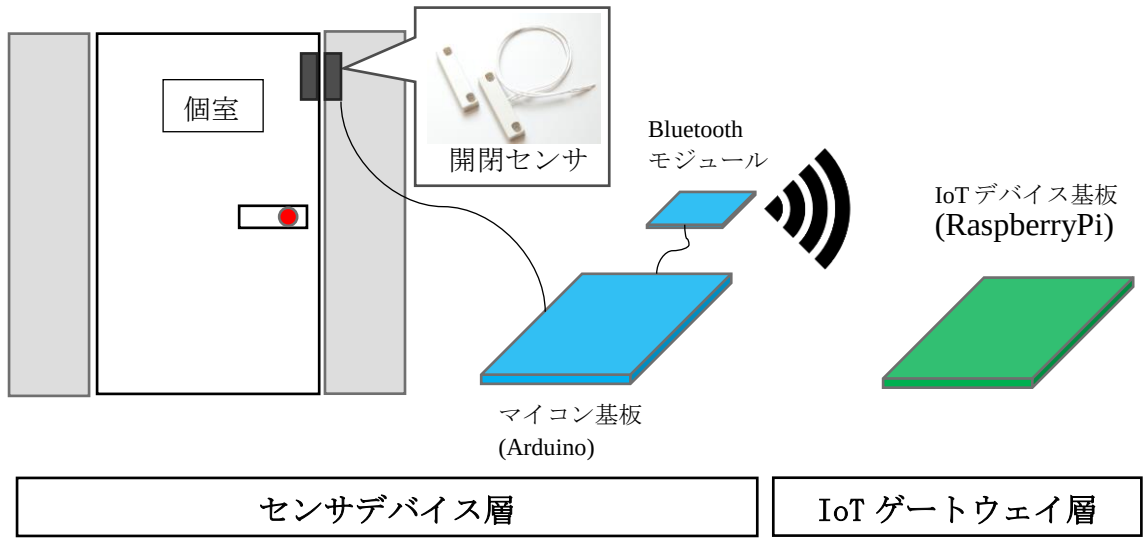


Bluetooth モジュール
通信範囲:30m 程度
通信速度:9600bps

Arduino
個室の利用状況の収集と
通信処理を実装

センサデバイス層の構成

センサデバイス層は、個室のドアに取り付けられた開閉検知の為の磁気リードスイッチから個室の利用状況を取得しその情報を Bluetooth 通信で IoT ゲートウェイ層のマイコン基板に送信するところまでを実装する。センサデバイス層のマイコン基板は、Arduino とし Bluetooth 通信は浅草ギ研が販売している Bluetooth モジュール BLEserial3 を利用し実現する。ここでは、Arduino の基本的な利用方法と基本的な入出力のプログラム、シリアル通信のプログラム、BLEserial3 の利用方法、実際のシステムで利用するセンサデバイス層の実装までを行う。



Arduino について

Arduino は、AVR マイコン※1 が搭載されたマイコンボードで入出力ポートを備えている。プログラミングは、C++風の言語で専用の統合開発環境を利用し行う。デジタル/アナログの入出力の関数はもちろんシリアル通信などの関数も既に準備されているので何かしらのプログラム言語経験者でプログラムの基本構文を理解している者にとっては簡単にプログラミングが可能である。

※1:Arduino の製品には、ARM を搭載した基板も存在する。

Arduino の製品ラインナップ

Arduino は、様々な種類の基板が販売されている。入出力ポートの数やシリアル通信のチャンネル数の違いがあったり、CAN 通信や LAN 通信を可能にした基板も存在する。価格も数百円のものから 9 千円台のものまであり用途に合わせて自由に選択できる。本実習では、Arduino Mega を利用する。ちなみに今回は簡単な入力処理とシリアル通信しか利用しないので基本的にはどの Arduino 基板でもよい。

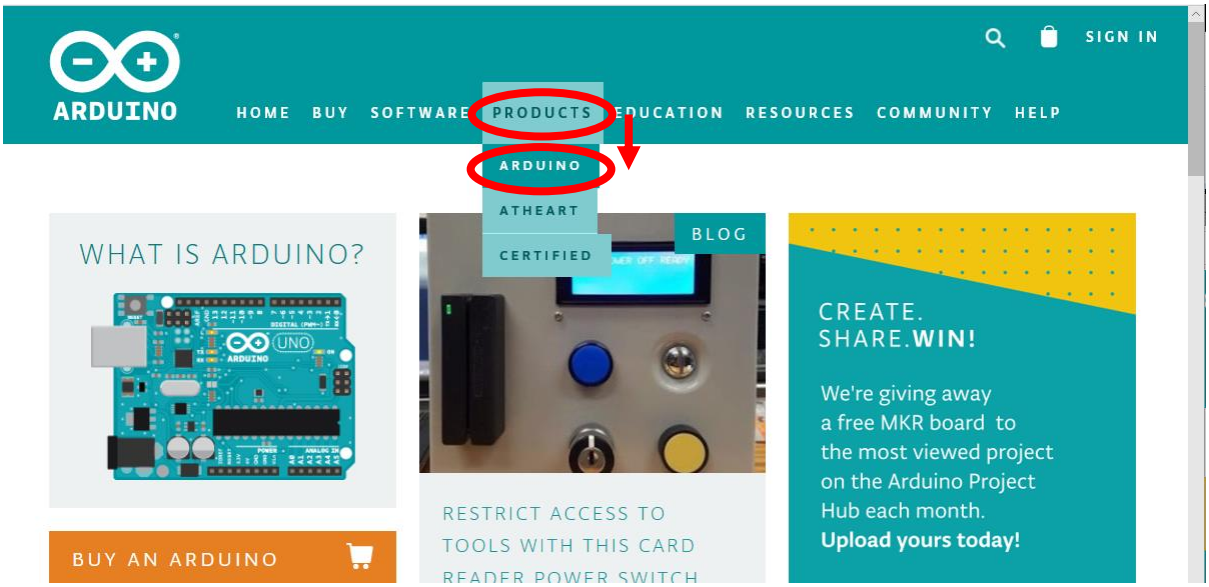
Arduino Mega の基板仕様

Arduino は、通常のマイコンのデータシートのような情報は公式サイトに掲載されている。このページには、基板の仕様だけではなくプログラムで利用する関数の説明やサンプルコードまで掲載されている。基本的には公式サイトの情報だけで自分が行いたい制御が実現できる。ここからは、Arduino Mega の基板を例に基板の仕様を公式サイトから確認する方法を示す。

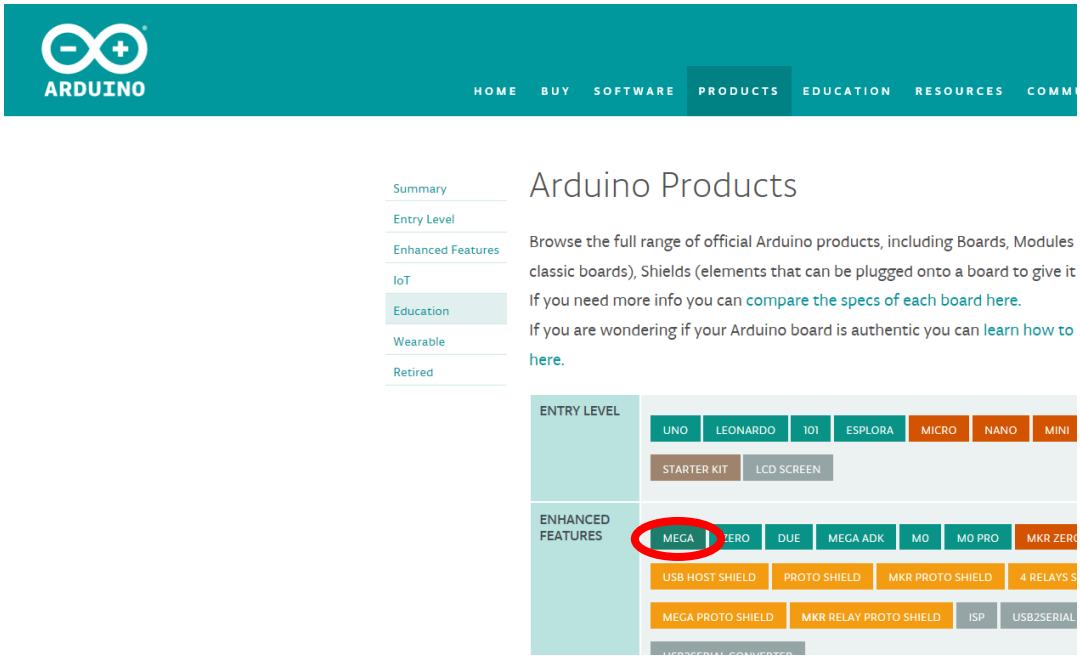
Arduino の公式サイト

https://www.arduino.cc/

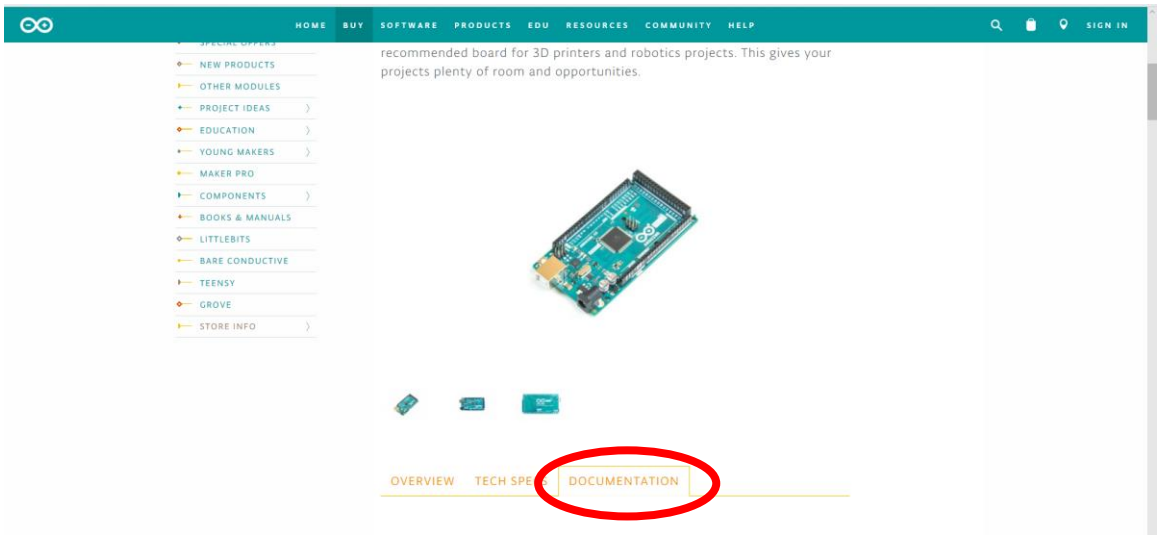
1．使用する基板の入出力ポートの構成を確認するには以下の「PRODUCTS」にカーソルを合わせると選択メニューが展開されるのでその中から「ARDUINO」をクリックする。



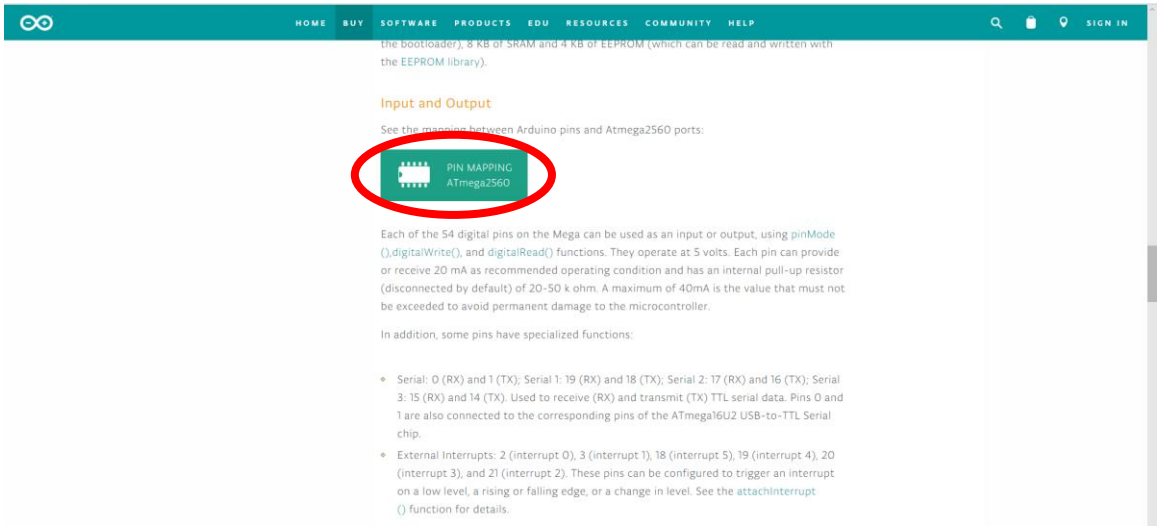
2．Arduino のラインナップ一覧が表示されるので使用する基板を選択する。例えば Arduino Mega を利用する場合は「Mega」をクリックする。



３．使用する基板の製品ページに飛ぶのでページをスクロールし中間あたりの「DOCUMENTATION」をクリックする。

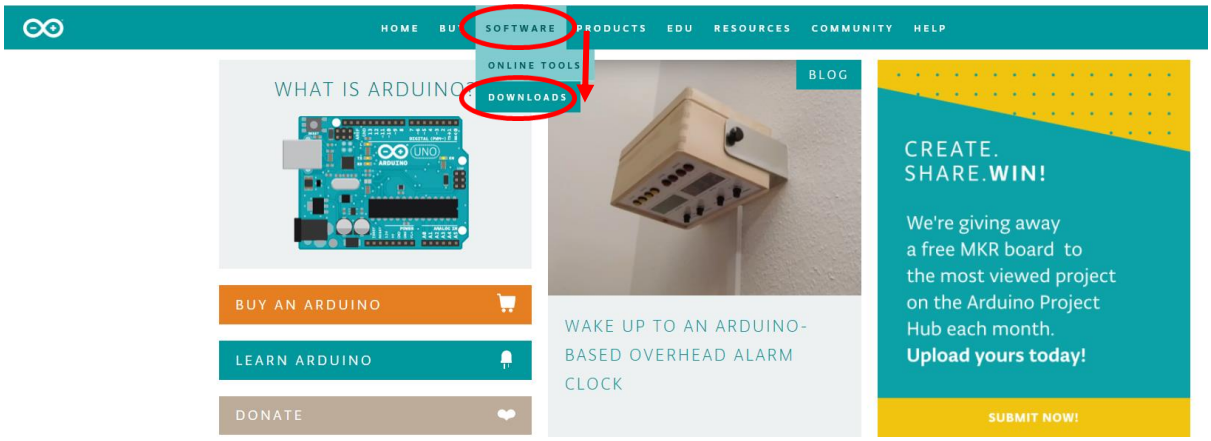


４．「DOCUMENTATION」をクリックすることで利用する基板の仕様を表示させることが出来る。例えば入出力ポートの構成を調べたい場合はスクロールし「Input and Output」の項目を確認する。入出力ポートの概要とピン配置を確認できる。



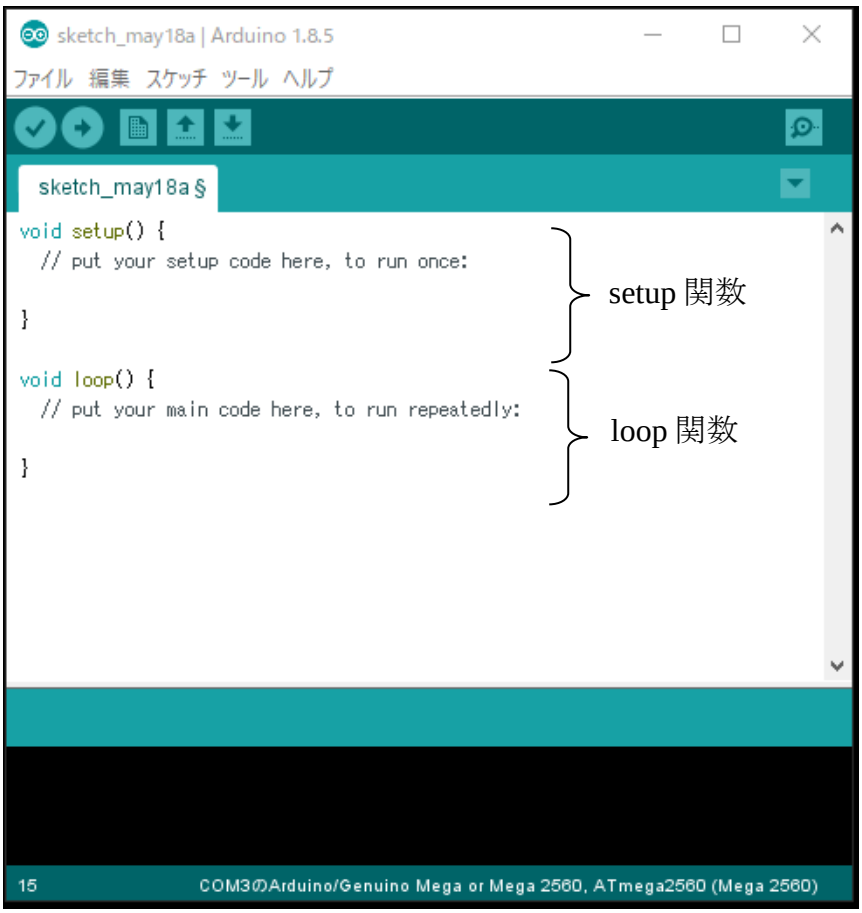
Arduino の統合開発環境

Arduino の開発は専用の統合開発環境を利用する。この開発環境は、公式サイトから簡単にダウンロードが出来る。公式サイトのトップページから「SOFTWARE」，「DOWNLOADS」を選択し手順に従ってダウンロードを行う。



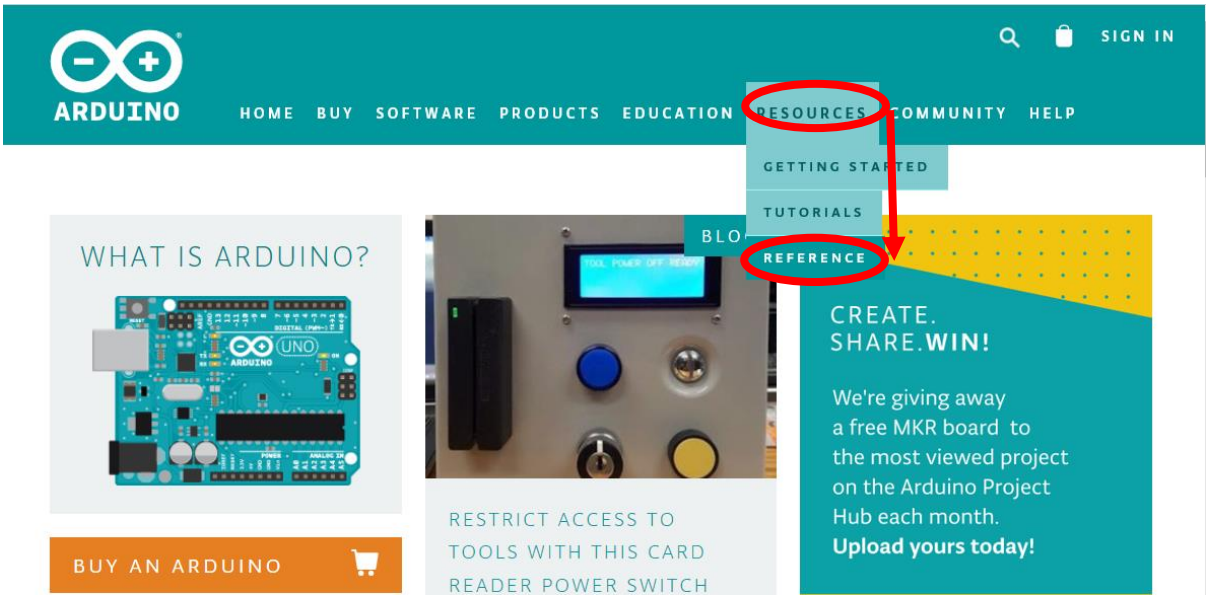
手順に従ってダウンロードするとデスクトップ上にショートカットが作成される。ダブルクリックすることで起動できる。

起動するとエディターが表示される。このエディターにプログラムを入力する。エディターが起動した時に既に「setup 関数」と「loop 関数」が入力されている。これら関数の詳細も公式サイトに掲載されている。

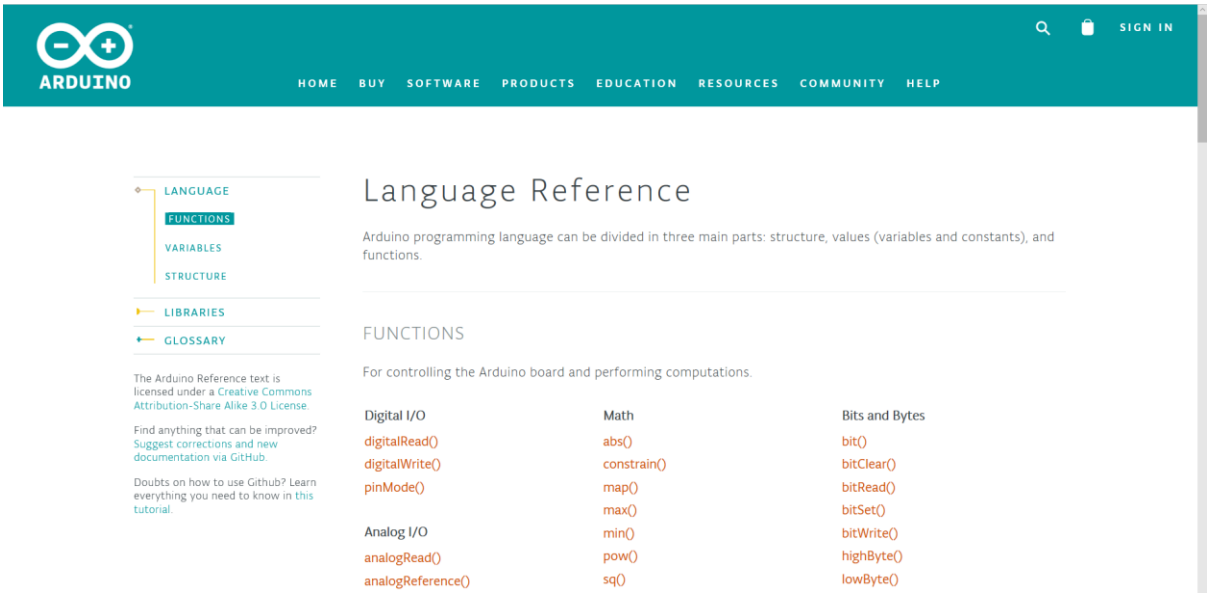


関数の詳細を確認する手順

公式サイトの「RESOURCES」にカーソルを合わせると選択メニューが表示される。その中から「REFERENCE」をクリックする。



表示されたページには各種関数の説明やその関数の利用例としてサンプルコード、データ型の説明、演算子の説明などプログラミングする上で必要となる基本的な情報を得ることが出来る。



例えば setup 関数の詳細を確認する。「Description」の部分が関数の説明となっている。

setup 関数の説明を要約すると以下となる。

> setup 関数は、初期化関数として利用し変数の初期化や入出力の設定等に使用する。
>
> Arduino 基板に電源を投入する又はリセットするたびに 1 回だけ実行される。

同様に loop 関数についても調べる。loop 関数の説明を要約すると以下となる。

> setup 関数が呼び出された後、loop 関数はその名の通り繰り返し処理を実行する。

つまり setup 関数は初期化関数、loop 関数は C 言語の while ループに相当する。これを踏まえ LED を点灯させるプログラムを作成する。

Arduino に LED を正論理接続し点灯させるには以下の手続きで行う。

①setup 関数内で LED を接続するピンを出力モードに設定する。

②loop 関数内で正論理で接続した LED を点灯させるために HIGH レベルの信号を出力する。

①,②を実装するために以下の関数を利用する。関数の説明は、先述した通り公式サイトで詳細を確認できる。

ピンの入出力モードを設定する関数

関数名	pinMode
説明	指定した pin を入力または出力として動作するように設定する
書式	pinMode(pin, mode)
引数	pin:指定するピン mode:INPUT または OUTPUT
戻り値	なし

出力ピンを制御する関数

関数名	digitalWrite
説明	指定した pin に HIGH または LOW を出力する
書式	digitalWrite(pin, value)
引数	pin:指定するピン value: HIGH または LOW
戻り値	なし

入出力モードを設定する pinMode 関数、出力を制御する digitalWrite 関数を利用し LED を点灯させるプログラムを作成する。以下はサンプルコードとなる。※1

サンプルコード(arduinoLED)

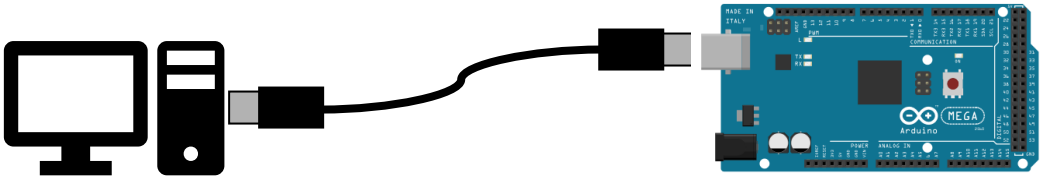
```
//30 番ピンを利用する場合
#define outpin1 30

void setup() {
  //30 番ピンを出力モードに設定
  pinMode(outpin1,OUTPUT);
}

void loop() {
  // HIGH レベルの信号を出力
  digitalWrite(outpin1, HIGH);
}
```

※1：出力ピンを 30 番ピンとした場合のサンプルコードとなる。必要に応じて変更すること。

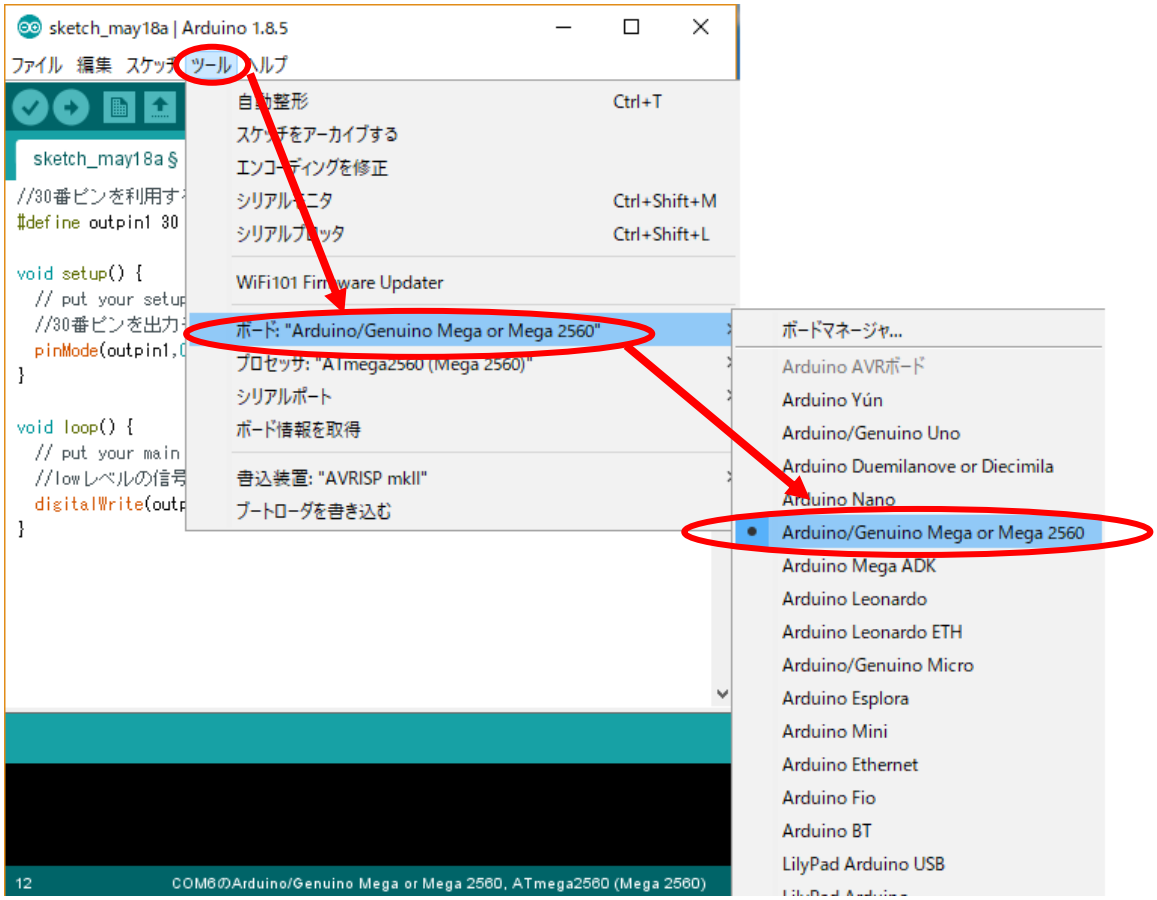
プログラムの書き込み

作成したプログラムは開発環境を利用し基板に対して書き込みを行えるようになっている。書き込み手順を以下に示す。

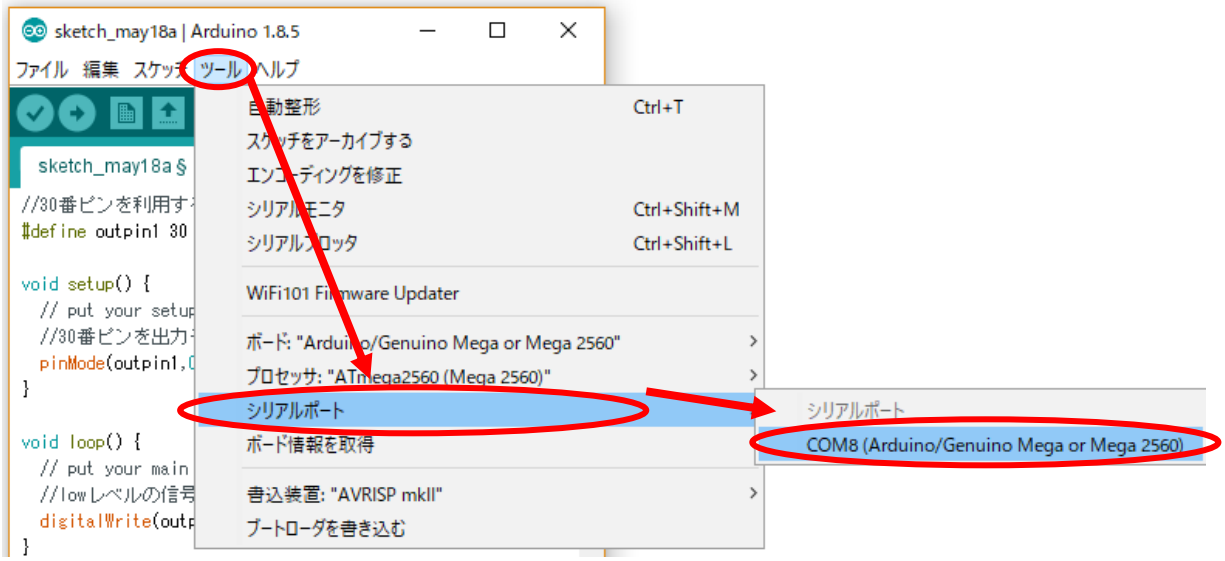
1. Arduino 基板とパソコンを USB ケーブルで接続する。



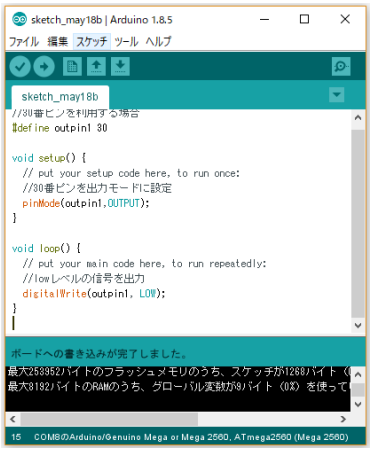
2. メニューバーの「ツール」から「ボード」にカーソルを合わせ表示される一覧から使用する Arduino 基板を決定する。



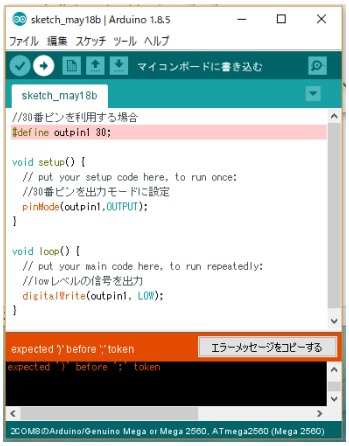
3. メニューバーの「ツール」から「シリアルポード」にカーソルを合わせ Arduino 基板が接続されている COM ポートを選択する。



4. 「マイコンボードに書き込む」アイコンをクリックし成功すると書き込みが完了のメッセージが表示される。エラーが出た場合はエラー内容が表示されるのでそれに従いプログラムを修正し再度書込みを行う。



書込み成功時



エラー時

Arduino プログラミング課題１

Arduino に１個の LED を接続し１秒周期(0.5 秒点灯し 0.5 秒消灯)で点滅させる。接続ピンは適当なものを選択する。

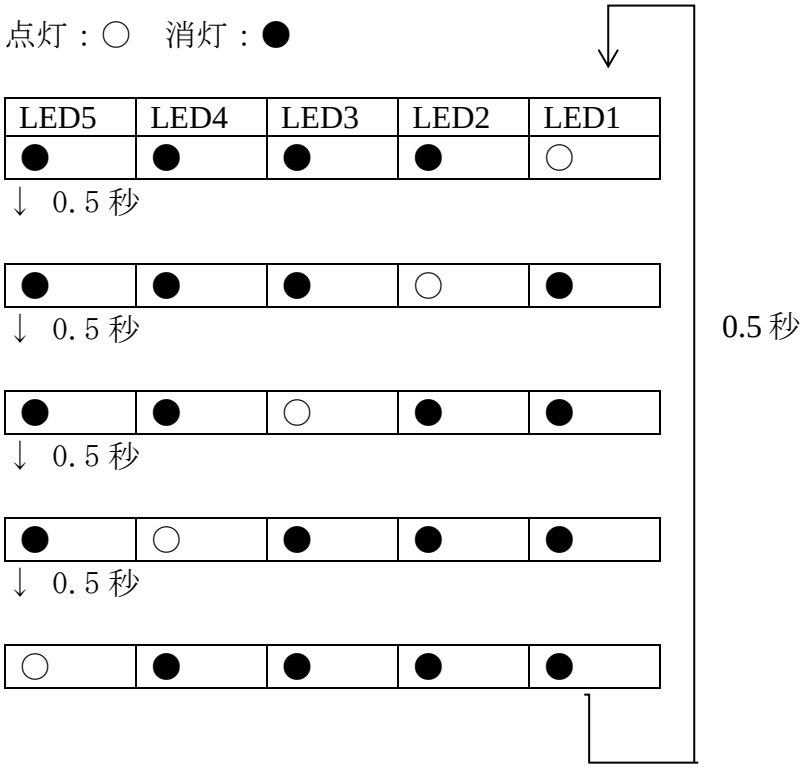
ヒント：時間を扱う関数を利用する

Arduino プログラミング課題２

Arduino に５個の LED を接続し以下のようにシフトさせながら点灯させる。接続ピンは適当なものを選択する。

ヒント：コードが冗長にならないように配列などを利用し工夫する

点灯：○　消灯：●



Arduino プログラミング課題３

Arduino に５個のトグルスイッチ(SW1 から SW5)と５個の LED(LED1 から LED5)を接続しそれぞれ対応するスイッチ(SW1 は LED1 等)の ON/OFF に合わせて LED を点灯/消灯させる。

ヒント：入力処理が必要になる。

シリアル通信のプログラム

Arduino はハードウェアシリアルとソフトウェアシリアルが利用できる。ハードウェアシリアルは、基板に搭載されているマイコンのハードとして持っているシリアル機能を利用するもので Arduino Mega は４チャンネル分利用できる。しかし、多くの Arduino 基板は 1 または 2 チャンネルのものが多い。そこで利用されるのがソフトウェアシリアルである。ソフトウェアシリアルは、本来ハードウェアで実現される機能をソフトウェアで実現するもので通常のデジタル入出力ピンをシリアル通信のピンとして使用できるものである。本教材では、ハードウェアシリアルだけを利用する。公式サイトでは Arduino Mega のハードウェアシリアル通信用のピンの説明が以下のようにされている。

Arduino Mega のシリアル通信ピンの仕様

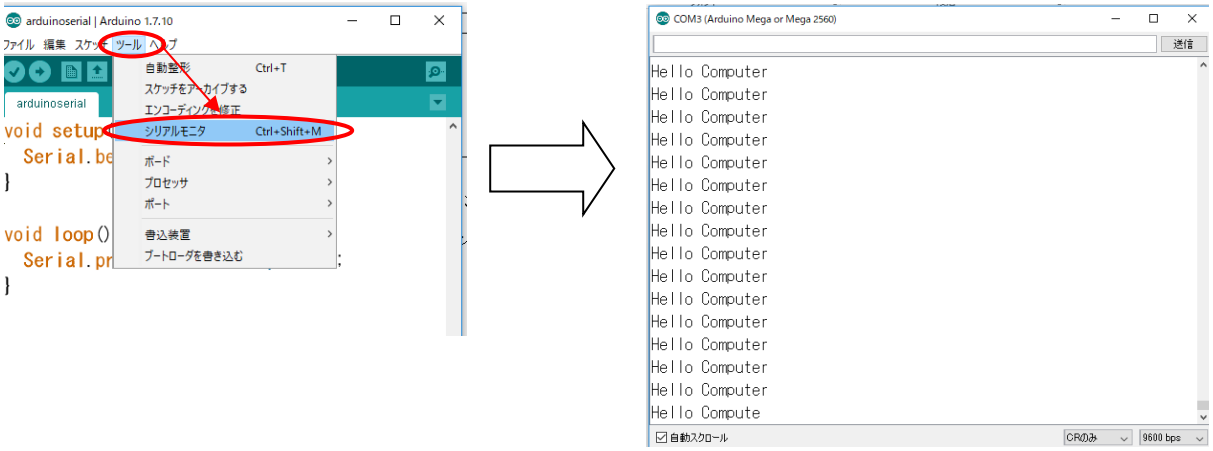
Arduino Mega には、通常の 0 番ピン(RX)と 1 番ピン(TX)の Serial に加え、19 番ピン（RX）と 18 番（TX）の Serial1、17 番（RX）と 16 番（TX）の Serial2、15 番（RX）と 14 番（TX）の Serial3 の 3 つの追加シリアルポートがある。

Arduino におけるシリアル通信のプログラムは簡単である。例として 0 番ピン(RX)と 1 番ピン(TX)の Serial ピンを利用しパソコンに文字列を送信するプログラムを作成する。以下がサンプルコードである。

サンプルコード

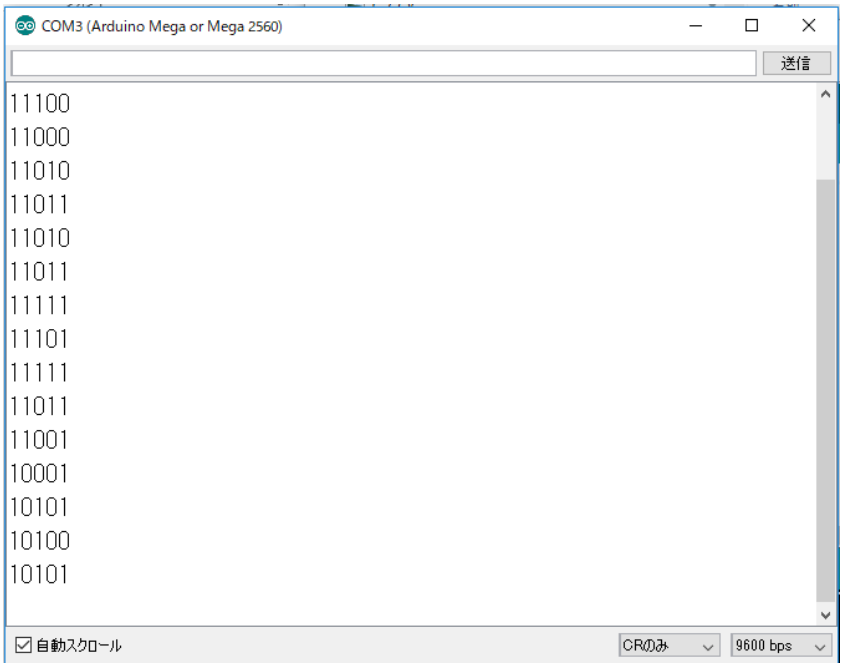
void setup() {
  Serial.begin(9600);
}
 	
void loop() {
  Serial.println("Hello Computer");
}

動作確認は開発環境に付属するシリアルモニタを利用し確認できる。Arduino とパソコンを USB ケーブルで接続した状態で開発環境のメニューバーの「ツール」から「シリアルモニタ」を選択する。シリアルモニタが起動し 1 番ピン(TX)から送信されている文字列をモニタすることができる。



Arduino プログラミング課題４

５個のトグルスイッチ(SW1~SW5)の ON/OFF の状態を１秒周期で送信する。この時、スイッチが ON の時「１」、OFF の時「０」とし５個のスイッチの ON/OFF の組み合わせデータとして送信する。送信するデータは Arduino 言語の String 型を利用し文字列データとして送信する。シリアルモニタを利用し５個のスイッチの状態をモニタリングすることで動作確認を行う。



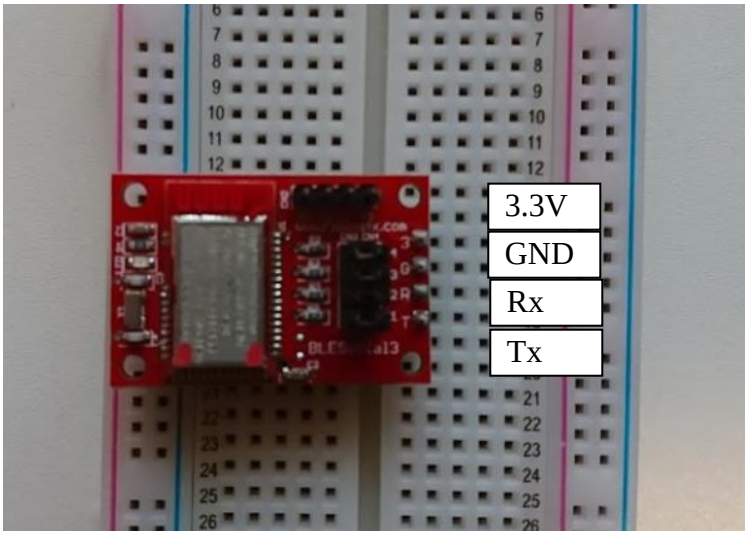
Arduino プログラミング課題５

課題４をスイッチの状況が変化したときにだけスイッチの状態を送信するように変更する。

ヒント：新しいスイッチの状態と前のスイッチの状態を比較し異なっていたら送信する。

Bluetooth モジュール

Bluetooth や Wi-Fi、Zigbee 等の無線通信は各種マイコンから簡単に利用できるようなモジュールが販売されている。モジュールを利用することで簡単に各種通信を実装することが出来る。今回は、無線通信として BluetoothLowEnergy(BLE)を利用することとしそれを実現するためのモジュールとして浅草ギ研が販売している BLEserial3 を利用する。通信に必要なペアリングなどの各種処理はモジュールに既に実装されており利用する側は単純なシリアル通信プログラムを実装するだけで BLE 通信を実現できる。このモジュールは、Tx、Rx、GND、3.3V の端子がある。モジュールの端子に Arduino の適当なピンを接続することで使用できる。

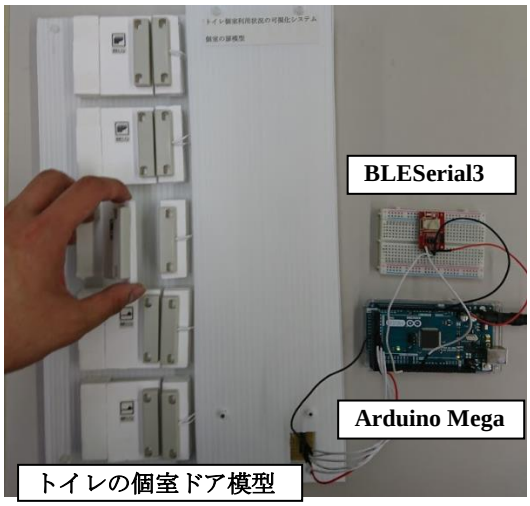


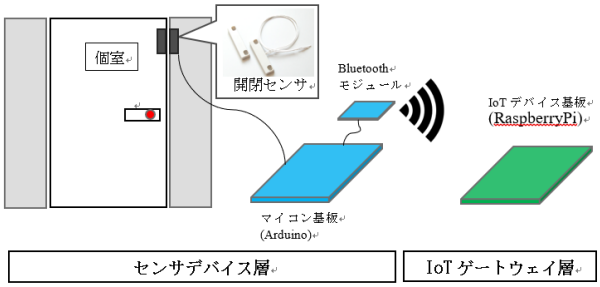
接続表

モジュール	Arduino
3.3V	3.3V
GND	GND
Rx	1 番ピン(TX)
Tx	0 番ピン(RX)

センサデバイス層の実装

個室の利用状況可視化システムに利用するセンサデバイス層部分の実装を行う。個室は５室と想定しそれぞれのドア部に開閉状態を検知するために磁気リードスイッチを取り付ける。磁気リードスイッチを Arduino に接続し５室分のドアの開閉状況を取得する。この開閉状況を BLEserial3 で IoT ゲートウェイ層に送信するところまでを実装する。



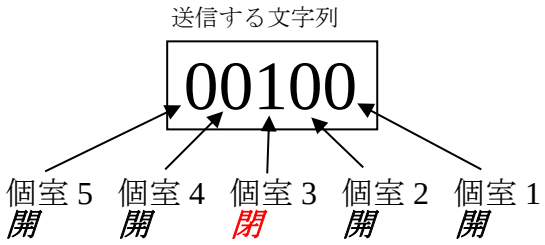


センサデバイス層の実装課題

磁気リードスイッチと BLEserial3 の接続は適当なピンを選択する。開閉状況は以下の仕様でパソコン側に送信する。また、送信は開閉状況が変化したときにだけ送信することとする。

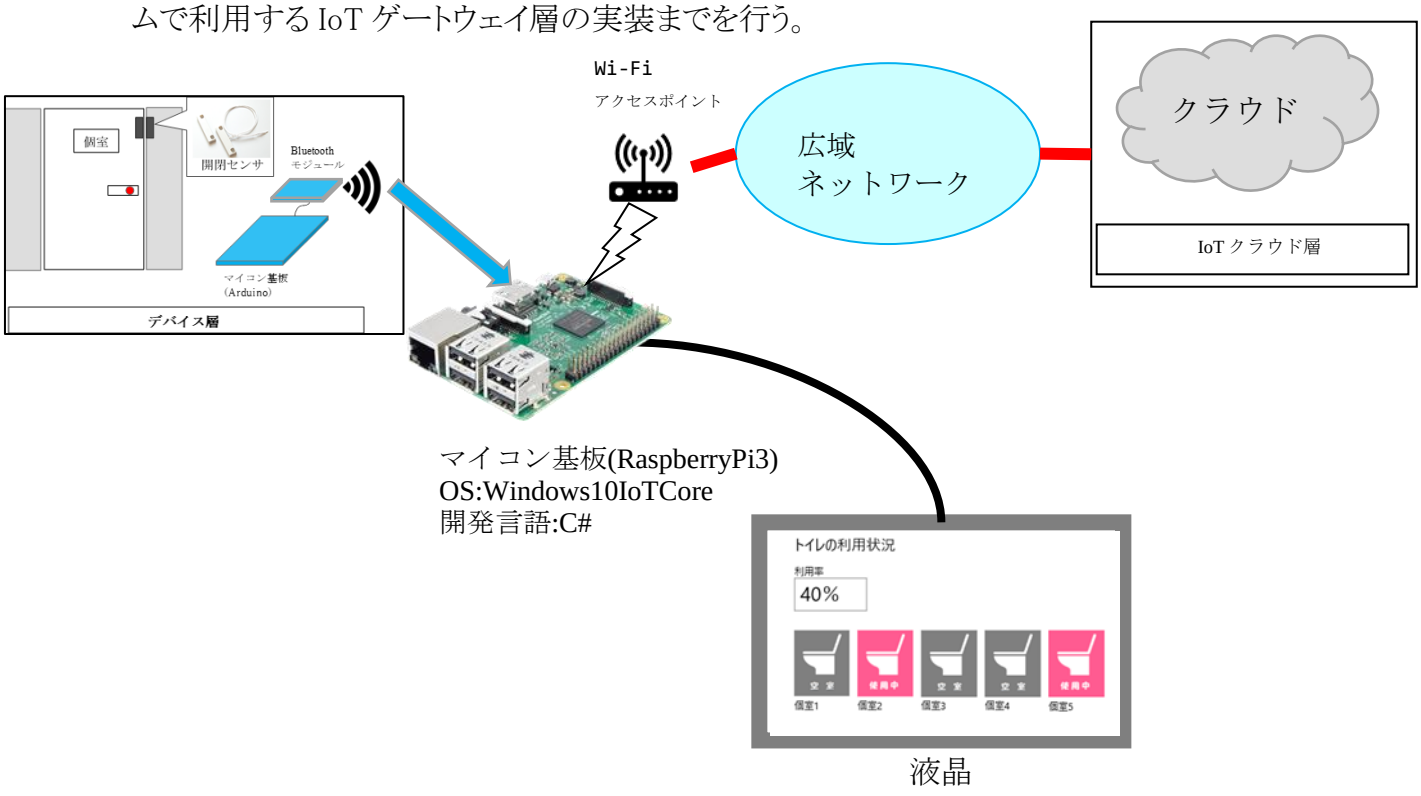
パソコン側に送信する際に開閉状況データ仕様

開いている状態の時「0」、閉まっている状態で「1」とし５室分の開閉状況を文字列で送信する。



IoT ゲートウェイ層の構成

IoT ゲートウェイ層は、センサデバイス層からのドアの開閉状況をもとに 2 つの処理を行う。1 つは、液晶ディスプレイに利用状況をリアルタイムに表示することで利用者に対し個室の利用状況を通知する処理、もう 1 つは広域ネットワーク経由でクラウドに利用状況を送信する処理である。これらの処理を行うマイコン基板は RspberryPi3 とする。RaspberryPi3 は、Bluetooth と Wi-Fi がオンボード上に搭載されていてセンサデバイス層からのデータ受信やクラウドへのデータ送信はこれらを利用することが出来る。RsapberryPi3 には、OS として Microsoft 社が提供する Windows 10IoTCore を搭載する。この OS で動作するアプリケーションは同じく Microsoft 社が提供する統合開発環境 Visual Studio を利用し C#等の言語で開発ができる。ここでは、アプリケーション作成の為の基本的なプログラミングの方法、Bluetooth 通信のプログラムの作成、クラウドサービスとの連携プログラムの作成を行い実際のシステムで利用する IoT ゲートウェイ層の実装までを行う。



IoT ゲートウェイ層

Windows10IoTCore

RaspberryPi3 には OS として Microsoft 社が提供する Windows10IoTCore を搭載する。この OS は、エンベデッドシステム用として提供されており Microsoft 社のホームページから無償でインストールできる。インストールの手順は公式のドキュメントが参考になる。

Windows10IoTCore のインストール手順

https://developer.microsoft.com/ja-jp/windows/iot/Downloads

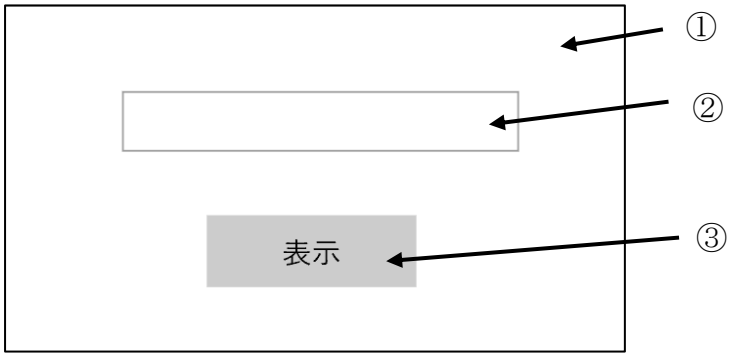
Windows10IoTCore 上で動作するアプリケーションは、統合開発環境 Visual Studio で開発できる。プログラミングから実際に作成したアプリケーションを RaspberryPi に配置するまでを Visual Studio で完結できる。開発言語は、C#や Visual Basic 等が利用できる。C#等を利用しデスクトップアプリケーションを作成したことがある人であれば戸惑うことなく開発ができる。開発に関する基本的な手順や詳しい情報は Microsoft 社のホームページが参考となる。

Windows10IoT の開発者向けガイド

https://docs.microsoft.com/en-us/windows/iot-core/

アプリケーションの作成

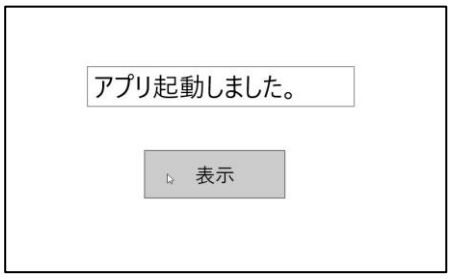
以下を参考に GUI を作成する。（プロジェクト名:btnmessage）



番号	コントロール種類	プロパティ項目名	プロパティ値
①	Grid	変更なし	変更なし
②	TextBox	名前	txtmsg
		Text	設定値無し
		FontSize	環境に合わせた適当な値
③	Button	名前	btnmsg
		Content	表示
		FontSize	環境に合わせた適当な値

アプリケーションの動作は以下とする。

① アプリケーション起動時にテキストボックスに「アプリ起動しました。」と表示する。



② 表示ボタンをクリックするとテキストボックスに「こんにちは」と表示する。

以下のイベントを登録する。

番号	対象コントロール	イベント名
①	btnmsg	Click

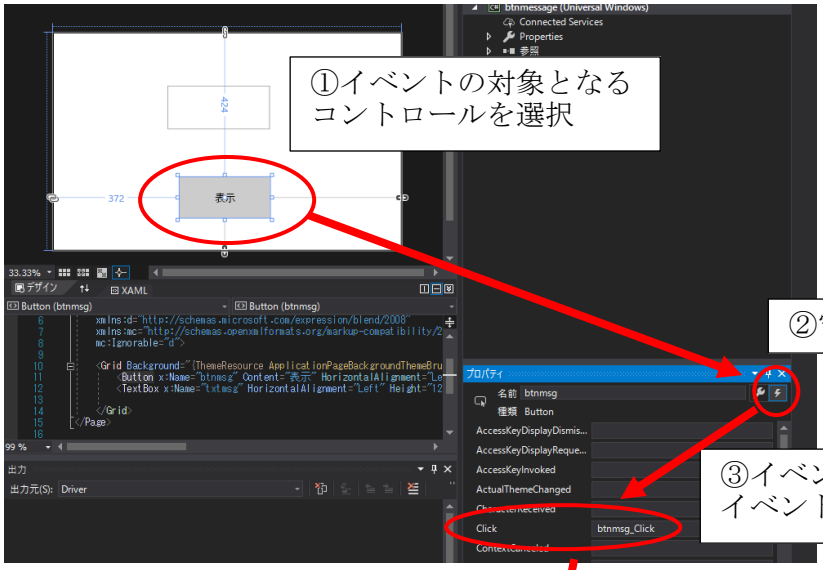
アプリケーションを作成する為のサンプルコードは以下となる。

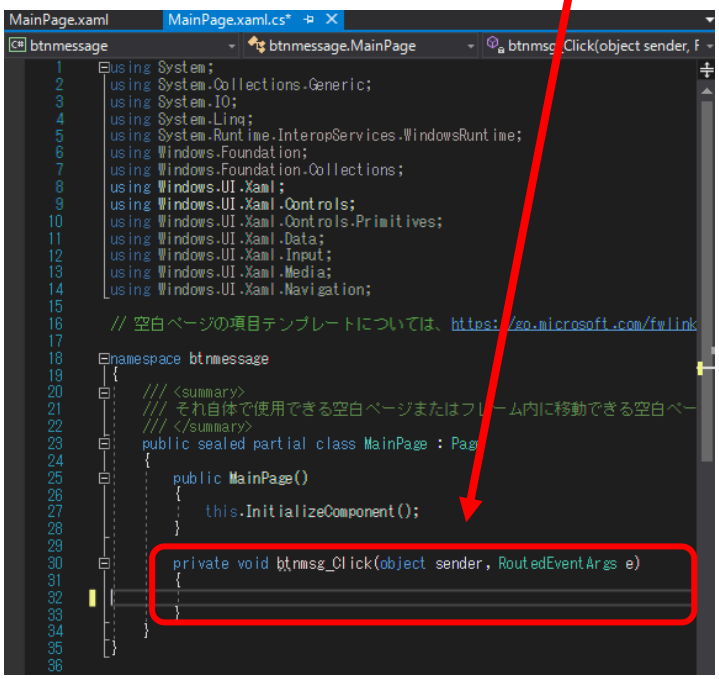
サンプルコード

```
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;


// 空白ページの項目テンプレートについては、
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x411 を参照してください

namespace btnmessage
{
    /// <summary>
    /// それ自体で使用できる空白ページまたはフレーム内に移動できる空白ページ。
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
            //テキストボックス(名前:txtmsg)に文字表示
            txtmsg.Text = "アプリ起動しました。";

        }

        private void btnmsg_Click(object sender, RoutedEventArgs e)
        {
            txtmsg.Text= "こんにちは";
        }
    }
}
```

イベント駆動型プログラム

ユーザーからの入力やデータの受信などのイベント発生を待って何らかの処理を実行するようなプログラムをイベント駆動型プログラムと呼ぶ。C#は、イベント駆動型のプログラムを簡単に記述できる構文が準備されている。GUI アプリケーションで利用する基本的なイベントの記述は、GUI を作成するデザイナーから自動生成することが出来る。





④コードエディターにイベントの記述が自動生成される。ここにイベント発生時に実行したい処理を記述する。

メソッドとプロパティ

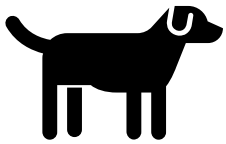
C#のプログラミング言語はオブジェクト指向と呼ばれる考えに基づいて作成されている。オブジェクト指向の最も基本的な仕組みがクラスである。オブジェクト指向の言語では、クラスを定義することによってプログラムを記述する。そのプログラムが動く時は、定義したクラスからオブジェクト(クラスから具象化した実体)が作られてそれらが相互作用しながらソフトウェアとしての機能を実現する。クラスでは、オブジェクトに対する操作(メソッド)とオブジェクトの属性(プロパティ)が定義されている。オブジェクト指向の考えでは、オブジェクトの中身がどうなっているのか(内部実装がどうなっているのか)は隠蔽されており必要な操作と属性だけを公開するという仕組みになっている。利用者はオブジェクトの内部実装がどのようになっているのかは意識せずにプログラミングが出来る。

定義されているクラス

```
//犬のクラス定義
class Dog
{
        //名前
        private string name;

        //コンストラクタ(インスタンス作成時に実行)
        Public void Dog(string name)
        {
                this.name = name;
        }

        //名前を取得できるプロパティ
        public string Name
        {
            get{ return this.name}
        }

        //「ワン」と吠えるメソッド
        public string cry()
        {
                return "ワン";
        }
}
```

クラス

内部実装



クラス名	Dog	
プロパティ	Name	名前を取得する
メソッド	cry()	ワンと鳴かせる

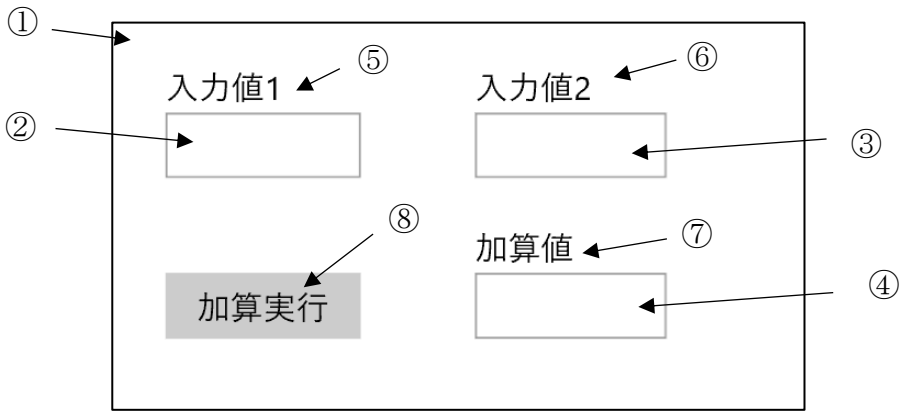
利用

クラスを利用する側は、内部実装がどのようになっているかは意識しないでプロパティやメソッドを利用して所望の処理が行える。

利用して作る

```
//犬クラスから2匹のインスタンスを生成
Dog pochi = new Dog("ポチ");
Dog taro = new Dog("タロー");
//メソッドを呼び出し吠えさせる
console.writeline(pochi.cry());
console.writeline(taro.cry());
```

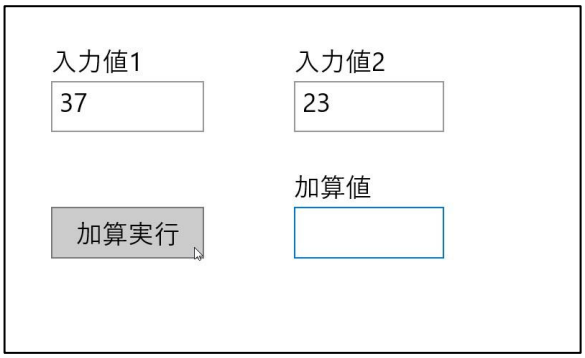
以下を参考に GUI を作成する。（プロジェクト名:numsum）



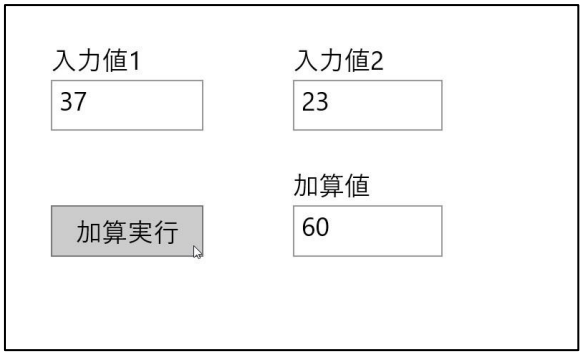
番号	コントロール種類	プロパティ項目名	プロパティ値
①	Grid	変更なし	変更なし
②	TextBox	名前	txtinpval1
		Text	設定値無し
		FontSize	環境に合わせた適当な値
③	TextBox	名前	txtinpval2
		Text	設定値無し
		FontSize	環境に合わせた適当な値
④	TextBox	名前	txtaddval
		Text	設定値無し
		FontSize	環境に合わせた適当な値
		IsReadOnly	true
⑤	TextBlock	名前	lblinpval1
		FontSize	環境に合わせた適当な値
		Text	入力値 1
⑥	TextBlock	名前	lblinpval2
		FontSize	環境に合わせた適当な値
		Text	入力値 2
⑦	TextBlock	名前	lbladdval
		FontSize	環境に合わせた適当な値
		Text	加算値
⑧	Button	名前	btnadd
		Content	加算実行
		FontSize	環境に合わせた適当な値

アプリケーションの動作は以下とする。

①入力値１と入力値２のテキストボックスに数値を入力する。

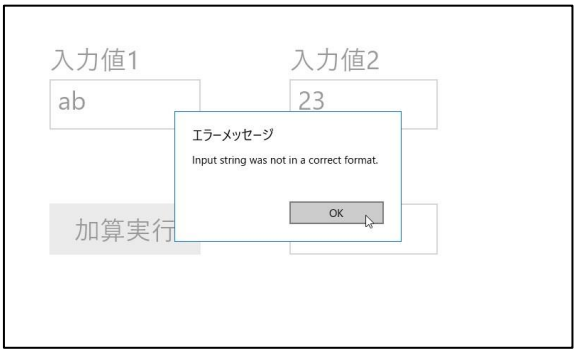


② 加算実行ボタンをクリックすると入力値 1 と入力値 2 の加算値が表示される。



③入力値として加算できない値を入力し加算を実行した場合はエラーであることを

メッセージボックスで通知する。



以下のイベントを登録する。

番号	対象コントロール	イベント名
①	btnadd	Click

アプリケーションを作成する為のサンプルコードは以下となる。

サンプルコード

```
using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

// 空白ページの項目テンプレートについては、
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x411 を参照してください

namespace numsum
{
    /// <summary>
    /// それ自体で使用できる空白ページまたはフレーム内に移動できる空白ページ。
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        private async void btnadd_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                txtaddval.Text = (Convert.ToInt32(txtinpval1.Text)
                                        + Convert.ToInt32(txtinpval2.Text)).ToString();
            }
            catch(Exception ex)
            {
                var msg = new ContentDialog();
                msg.Title = "エラーメッセージ";
                msg.Content = ex.Message;
                msg.PrimaryButtonText = "OK";
                await msg.ShowAsync();
            }
        }
    }
}
```

構造化例外処理

Convert.ToInt32 メソッドは引数で指定した値を 32 ビット符号付き整数に変換する処理を実行する。今回は、テキストボックスに入力された String 型の値を整数に変換する為にメソッドを利用している。このメソッドには実行時に例外となる処理が定義されている。Convert. ToInt32 メソッドを実行するとき引き渡された文字列が 0～9 の数値表現ではなく、例えば、「A」や「あ」など Int32 値に変換出来ないようなものが渡った時、変換できないので例外オブジェクトをスローする仕組みが設けられている。これは、このメソッドに限ったことではなく他のメソッドも例外時に例外オブジェクトをスローするものが存在する。開発者は予め使用するメソッドの例外について確認しておく必要がある。例外処理の構文を以下に示す。例外が発生するメソッドを使用する場合は try ブロックに処理を記述する。もし、例外が発生したら catch ブロックに処理が移り例外の処理を施す。ここでは、例外の処理の他に例外の発生をユーザーに知らせたりする必要がある。finally ブロックは、例外発生の有無にかかわらず実行すべき処理を記述する。finally ブロックは、省略可能である。

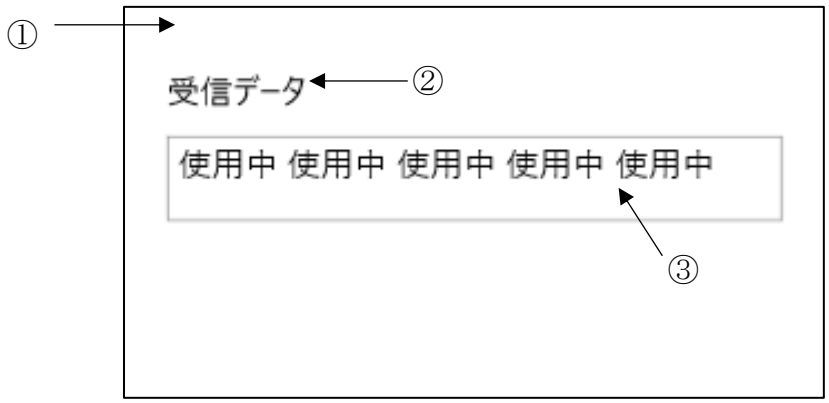
```
try
{
  例外が発生する可能性のある処理を記述する。
}
catch(例外クラス 変数)
{
  例外が発生した際の処理
}
finally
{
  例外の有無に関係なく最後にかならず処理
}
```

Bluetooth 通信のプログラム

Microsoft 社は Bluetooth デバイスとの通信を可能とするクラスライブラリを提供している。Bluetooth 通信のプログラムは、そのクラスライブラリを利用し実装する。Bluetooth に関するクラスライブラリの利用方法は Microsoft 社のドキュメントを参考にすると良い。今回は以下のドキュメントを参考にサンプルコードを作成している。

https://docs.microsoft.com/en-us/windows/uwp/devices-sensors/gatt-client

Bluetooth デバイスとの通信を確立しデータを受信するプログラムを作成する。以下を参考に GUI を作成する。（プロジェクト名:Bluetoothsample）

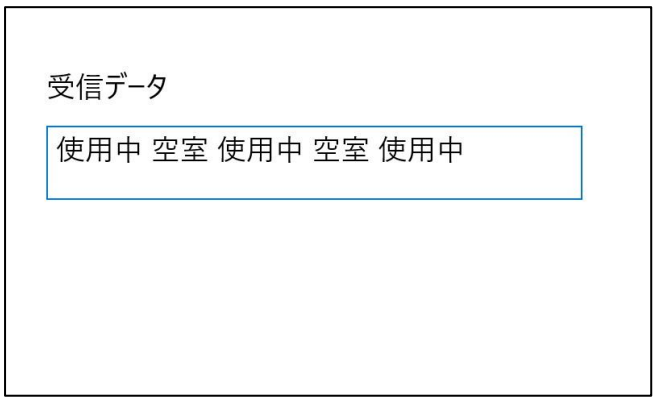


番号	コントロール種類	プロパティ項目名	プロパティ値
①	Grid	変更なし	変更なし
②	TextBlock	名前	lblrexeivedata
		FontSize	環境に合わせた適当な値
		Text	受信データ
③	TextBox	名前	txtrexeivedata
		Text	使用中 使用中 使用中 使用中 使用中
		FontSize	環境に合わせた適当な値
		IsReadOnly	true

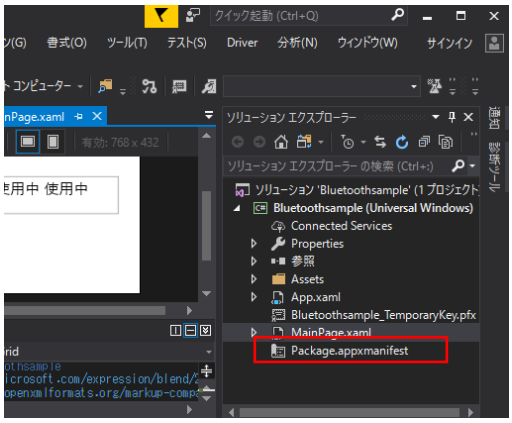
以下のイベントを登録する。

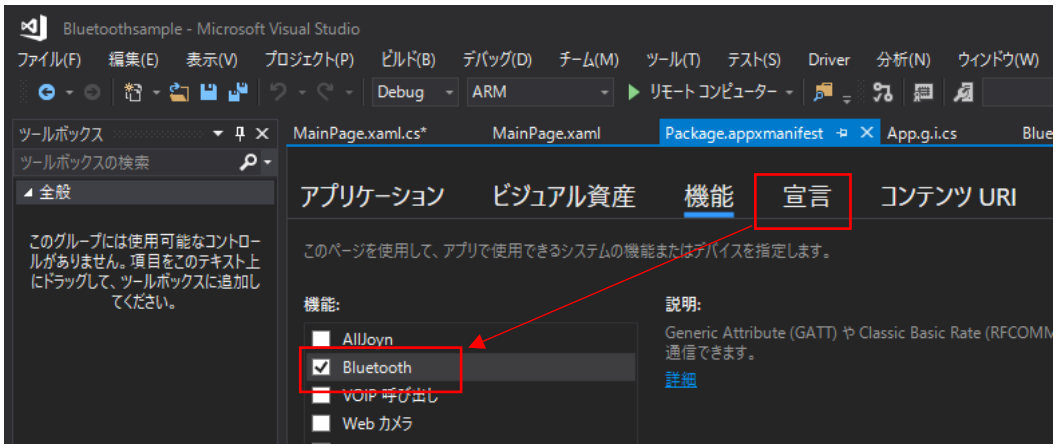
番号	対象コントロール	イベント名
①	Grid	Loeded

アプリケーションの動作は、既に作成したセンサデバイス層と通信を確立し個室の利用状況データを受信する。受信データを利用し個室の利用状況をリアルタイムにテキストボックスに表示する。



アプリケーションで Bluetooth を利用するためにはプロジェクト作成後にソリューションエクスプローラーから Package.appxmanifest をダブルクリックし機能タブを選択する。Bluetooth にチェックを入れる必要がある。





サンプルコードは以下となる。

サンプルコード

```
using System;
using System.Linq;
using Windows.Devices.Bluetooth;
using Windows.Devices.Bluetooth.GenericAttributeProfile;
using Windows.Devices.Enumeration;
using Windows.Storage.Streams;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

// 空白ページの項目テンプレートについては、https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x411 を参照してください

namespace Bluetoothsample
{
    /// <summary>
    /// それ自体で使用できる空白ページまたはフレーム内に移動できる空白ページ。
    /// </summary>
    public sealed partial class MainPage : Page
    {
        //Bluetooth
        //使用するデバイスで値を設定
        private Guid SERVICE_UUID = new Guid("FEED0001-C497-4476-A7ED-727DE7648AB1");
        //使用するデバイスで値を設定
        private Guid CHARACTERSTI_UUID = new Guid("FEEDAA03-C497-4476-A7ED-727DE7648AB1");
        private DeviceInformationCollection deviceInfos;
        private GattCharacteristicsResult gattCharacteristics;

        public MainPage()
        {
            this.InitializeComponent();
        }

        private void Grid_Loaded(object sender, RoutedEventArgs e)
        {
            this.BluetoothAsync();
        }

        //Bluetooth接続処理
        private async void BluetoothAsync()
        {
            37ページのBluetoothの接続処理のコードを入力
        }
        //データの受信イベント
        private async void GattCharacteristicChange(GattCharacteristic sender, GattValueChangedEventArgs args)
        {
            37ページのデータ受信イベントのコードを入力
       }
     }
}
```

Bluetooth の接続処理

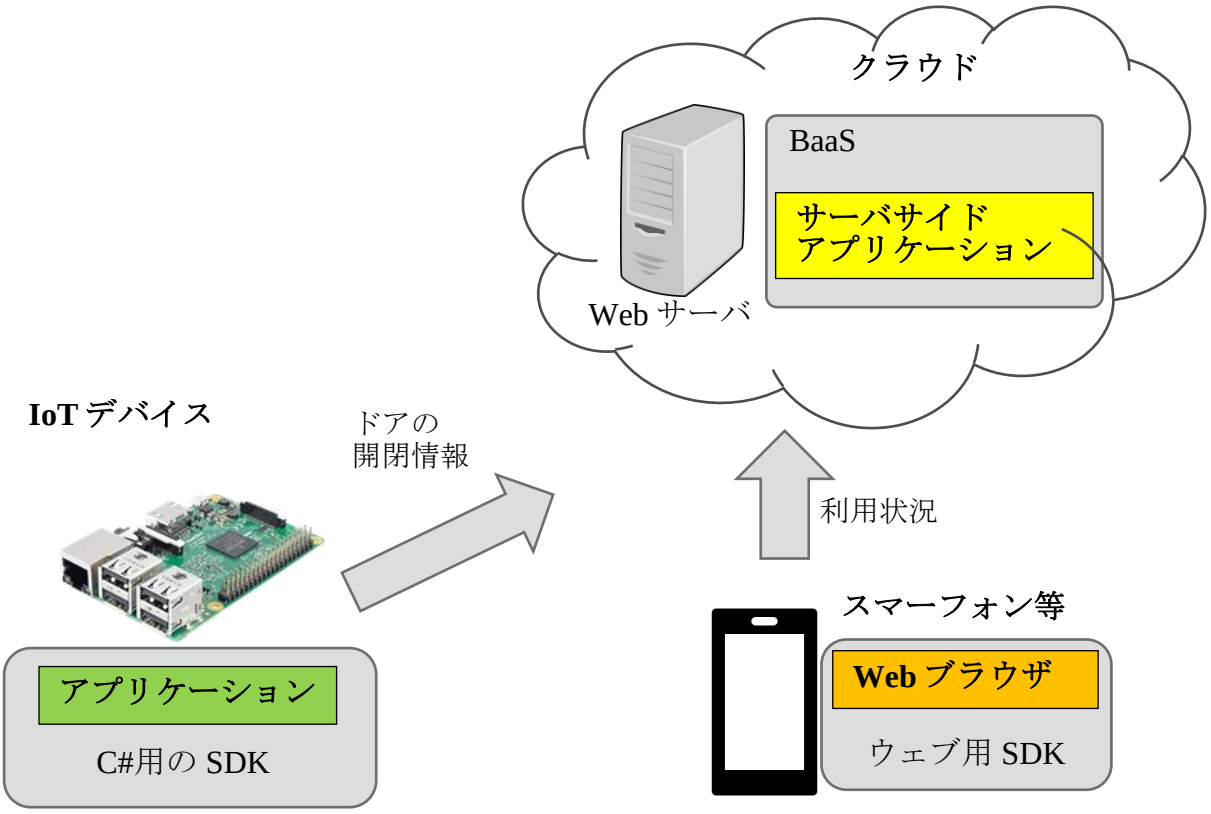
```
//Bluetooth接続処理
private async void BluetoothAsync()
{
    try
    {
        deviceInfos = await DeviceInformation.FindAllAsync(BluetoothLEDevice.GetDeviceSelectorFromDeviceName("BLESerial_0"));

        if (deviceInfos == null || deviceInfos.Count != 1) return;

        BluetoothLEDevice bleDevice = await BluetoothLEDevice.FromIdAsync(deviceInfos.First().Id);

        GattDeviceServicesResult gattDeviceServices = await bleDevice.GetGattServicesForUuidAsync(SERVICE_UUID);

        if (gattDeviceServices.Status == GattCommunicationStatus.Success)
        {
            gattCharacteristics = await gattDeviceServices.Services.First().GetCharacteristicsForUuidAsync(CHARACTERSTI_UUID);

            if (this.gattCharacteristics.Status == GattCommunicationStatus.Success)
            {

                var gattCharacteristic = this.gattCharacteristics.Characteristics.First();

                GattCommunicationStatus status = await gattCharacteristic.WriteClientCharacteristicConfigurationDescriptorAsync
                                                                    (GattClientCharacteristicConfigurationDescriptorValue.Notify);
                if (status == GattCommunicationStatus.Success)
                {
                    gattCharacteristic.ValueChanged += GattCharacteristicChange;
                }
            }
        }
    }
    catch (Exception ex)
    {
        var msg = new ContentDialog();
        msg.Title = "エラーメッセージ";
        msg.Content = ex.Message;
        await msg.ShowAsync();
    }
}
```

データ受信イベント

```
//データの受信イベント
private async void GattCharacteristicChange(GattCharacteristic sender, GattValueChangedEventArgs args)
{


    byte[] bArray = new byte[args.CharacteristicValue.Length];
    DataReader.FromBuffer(args.CharacteristicValue).ReadBytes(bArray);

    //別スレッドからのUI操作処理
    await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, async () =>
    {
        try
        {
            for (int i = 0; i < bArray.Length; i++)
            {
                if (bArray[i] == '1')
                {
                    strdata.Append("使用中 ");
                }
                else
                {
                    strdata.Append("空室 ");
                }
            }

            txtrexeivedata.Text = strdata.ToString();
            strdata.Clear();
        }
        catch (Exception ex)
        {
            var msg = new ContentDialog();
            msg.Title = "エラーメッセージ";
            msg.Content = ex.Message;
            await msg.ShowAsync();
        }
    });
}
```

Backend as a service(BaaS)の利用

BaaS は、Web アプリケーションに必要なサーバサイドの機能を提供するクラウドサービスである。BaaS を利用することでサーバサイド側を開発することなく Web アプリケーションを構築することが出来る。またサーバサイド側の機能を利用するための SDK(Software Development Kit)も提供されており利用者はそれらをシステム内に組み込みアプリケーション側からクラウドサービスを利用することが出来る。BaaS は、いくつかのサービスが存在するが今回は Google 社が提供する Firebase を利用する。



Firebase について

Firebase は、Google 社が提供する BaaS である。BaaS は、クラウドサービスであるため基本的には利用した分だけの料金が発生するが Firebase の場合は無料枠があり実験程度であればこの無料枠の中でも十分に使える。通常の Google アカウントが利用できるためクレジットカードの登録など余計な手続きは必要としない。Firebase をアプリケーションから利用するための SDK は、iOS アプリ用,Android アプリ用,ウェブ用等が公式にサポートされている。また公式ではないが C#をはじめ他の言語で利用できる SDK も存在する。

Firebase 公式サイト

https://firebase.google.com/

Firebase にログインするとユーザー専用のトップページが表示される。新しいプロジェクトを追加するために「プロジェクトを追加」をクリックし、プロジェクト名、国、同意の為のチェックを入れ「プロジェクトを作成」をクリックする。

プロジェクトを作成することでプロジェクトの管理ページが表示される。BaaS として様々な機能を利用することが出来るが今回は Database と Hosting のサービスだけを利用する。

Database の機能は、データの保管や同期を行うものだが私たちが一般的にイメージするリレーショナルデータベースとは少し異なる。今回使用する Database の機能は正式には Firebase Realtime Database という名称である。Firebase Realtime Database 公式には以下のように説明されている。

説明抜粋

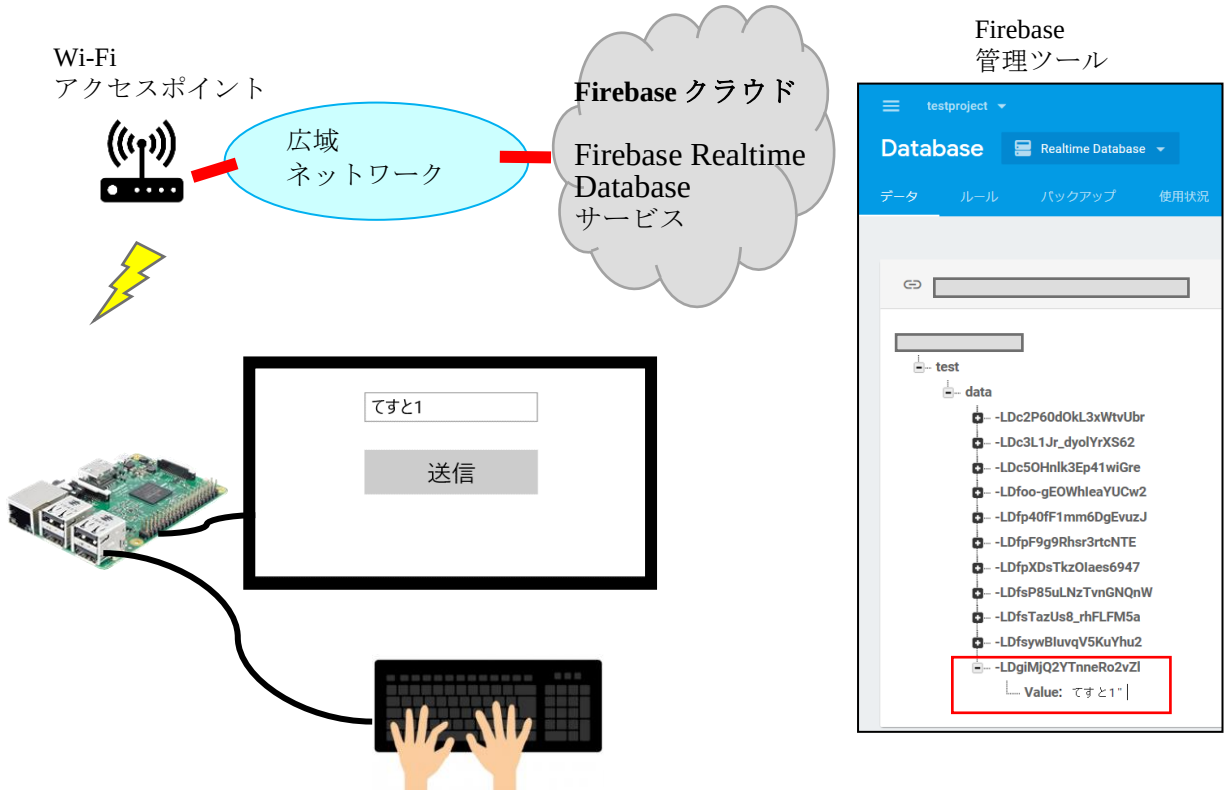
> NoSQL クラウド データベースでデータの保管と同期を行うことができます。データはすべてのクライアントにわたってリアルタイムで同期され、アプリがオフラインになっても、利用可能な状態を保ちます。Firebase Realtime Database はクラウドでホスティングされるデータベースです。データは JSON として保存され、接続されているすべてのクライアントとリアルタイムで同期されます。

Firebase Realtime Database へのデータ保存

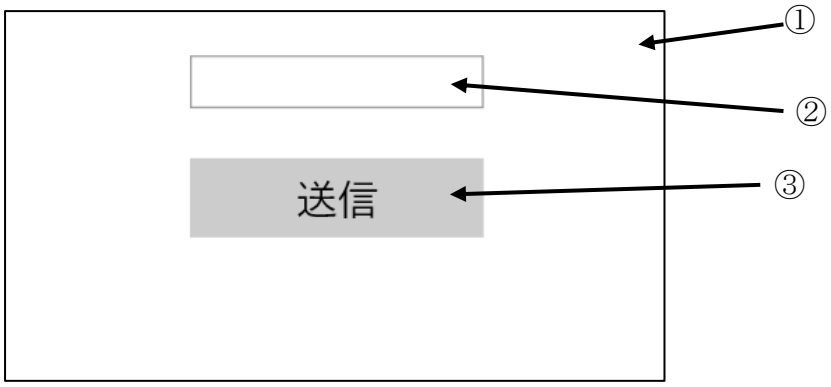
以下の機能を持ったアプリケーションを作成し Firebase との連携の基礎を習得する。

①テキストボックスに文字列を入力し送信ボタンをクリックする。

②入力した文字列が Firebase Realtime Database へ保存される。保存された文字列は管理ツールから確認できる。



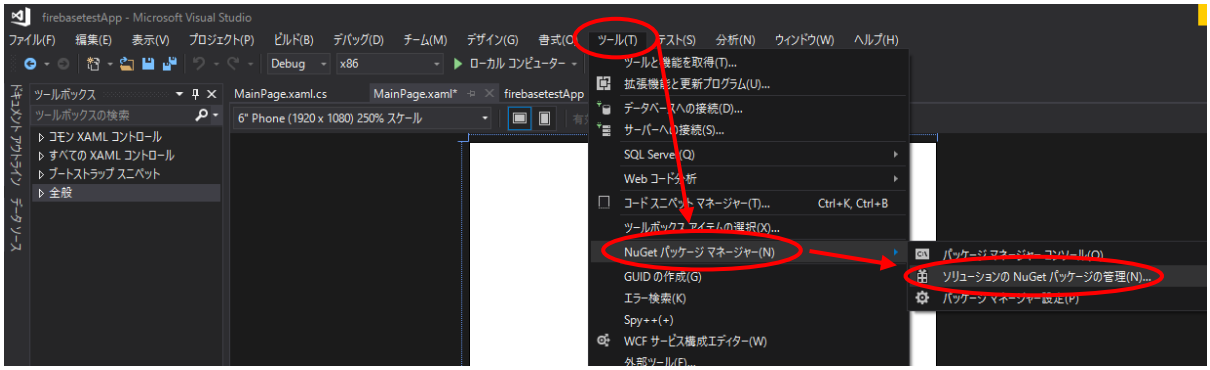
以下を参考に GUI を作成する。（プロジェクト名:firebasetestApp）



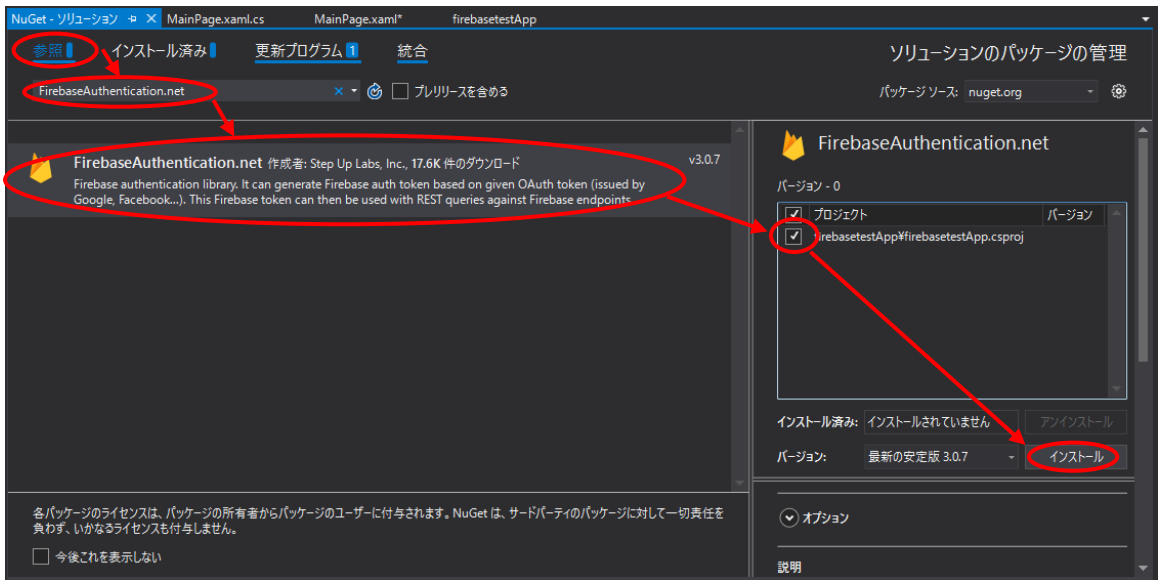
番号	コントロール種類	プロパティ項目名	プロパティ値
①	Grid	変更なし	変更なし
②	TextBox	名前	txtdata
		Text	設定値無し
		FontSize	環境に合わせた適当な値
③	Button	名前	btnsend
		Content	送信
		FontSize	環境に合わせた適当な値

アプリケーション側から Database にログインできる適当なユーザーを作成する。ブラウザから管理ツールにアクセスし「Authentication」を選択する。表示される画面から「ユーザーを追加」をクリックし入力画面に「メールアドレス」、「パスワード」を入力し「ユーザーを追加」をクリックする。今回は適当なユーザーを作成すればいいのでメールアドレスを ***test@example.com*** としパスワードは ***password*** とする。

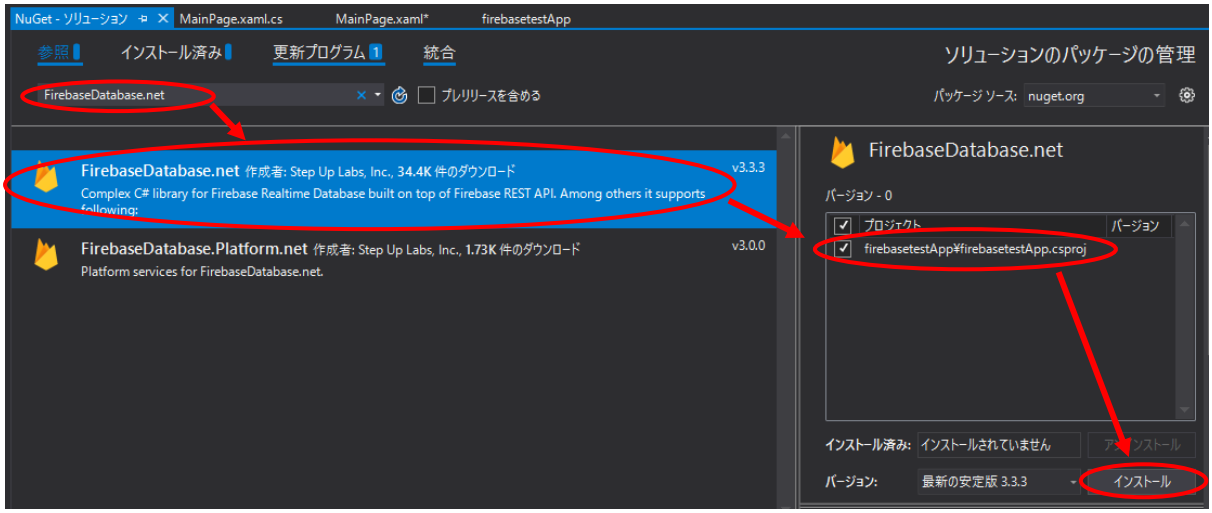
Firebase をアプリケーション側から利用するために C#用の SDK をインストールする。Visual Studio のメニューバーの「ツール」から「NuGet パッケージマネージャー」を選択し「ソリューションの NuGet パッケージの管理」をクリックする。NuGet は、Microsoft 社が提供するプログラムを共有する仕組みで他の開発者がプログラムを開発し公開している場合、それらを簡単に作成しているプロジェクトに取り込むことができる。



Nuget の管理画面が表示されるので「参照」タブをクリックし検索欄に *FirebaseAuthentication.net* と入力する。今回必要となる C#用 SDK のひとつが検索されるのでインストールの対象のプロジェクトにチェックを入れ「インストール」をクリックする。



同じく検索欄に *FirebaseDatabase.net* と入力しインストールを行う。



インストールした *FirebaseAuthentication.net* と *FirebaseDatabase.net* は Firebase をアプリケーション側から利用するための C#用の SDK である。この SDK は有志の方が作成し公開しているもので SDK の利用方法やサンプルプログラムは以下で確認できる。

FirebaseAuthentication.net の詳細

https://github.com/step-up-labs/firebase-authentication-dotnet

FirebaseDatabase.net の詳細

https://github.com/step-up-labs/firebase-database-dotnet

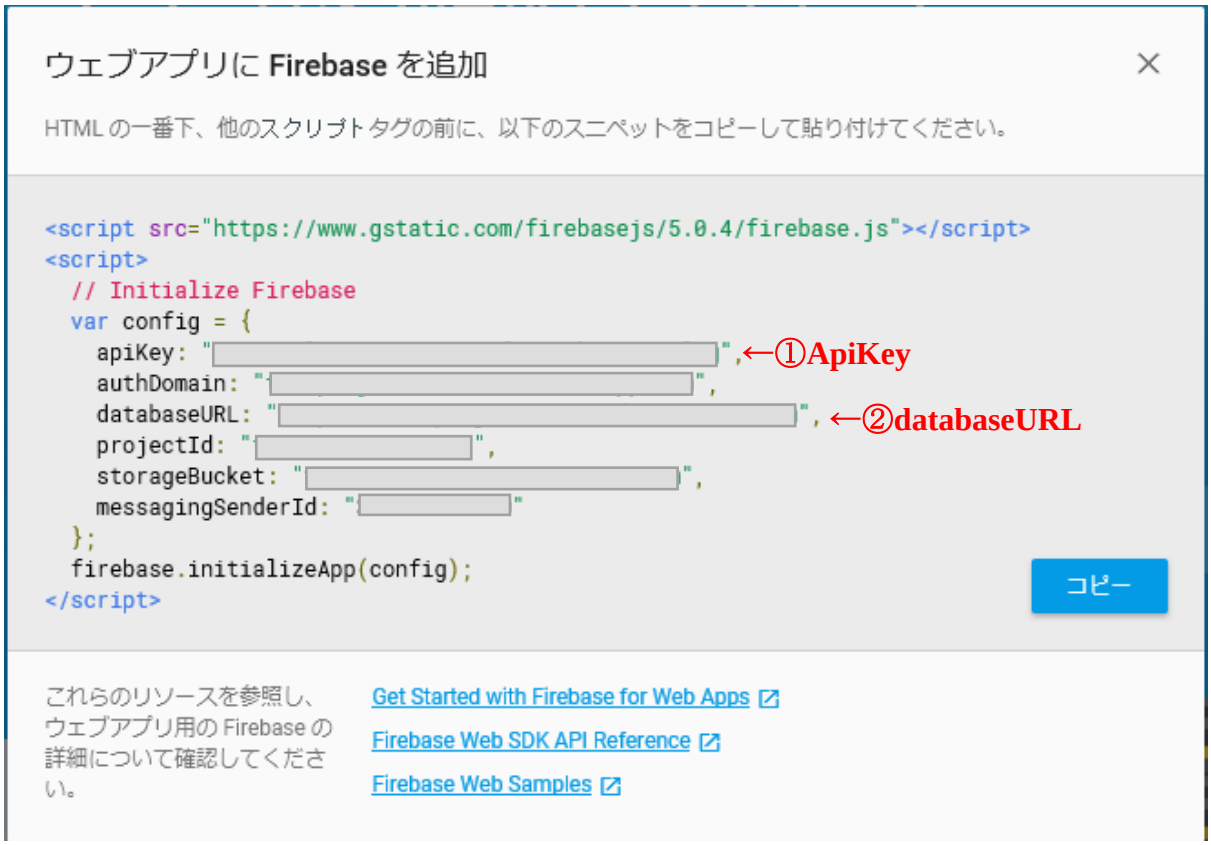
サンプルコードは以下となる。

サンプルコード

```
using System;
using System.Diagnostics;
using System.Threading.Tasks;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;


using Firebase.Auth;
using Firebase.Database;
using Firebase.Database.Query;

// 空白ページの項目テンプレートについては、
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x411 を参照してください

namespace firebasetestApp
{
    /// <summary>
    /// それ自体で使用できる空白ページまたはフレーム内に移動できる空白ページ。
    /// </summary>
    public sealed partial class MainPage : Page
    {
        private static string ApiKey = "          ①48ページ参照          ";
        public const string databaseURL = "          ②48ページ参照          ";
        private static string AuthEmail = "test@example.com";
        private static string AuthPassword = "password";

        private FirebaseAuthLink authLink;

        public MainPage()
        {
            this.InitializeComponent();
            this.firebaseconnect();
        }

        private async void btnsend_Click(object sender, RoutedEventArgs e)
        {
            await sendRun(authLink,txtdata.Text);
        }
        //Firebaseとの接続処理
        private async void firebaseconnect()
        {
            try
            {
                var auth = new FirebaseAuthProvider(new FirebaseConfig(ApiKey));
                authLink = await auth.SignInWithEmailAndPasswordAsync(AuthEmail, AuthPassword);
            }
            catch (Exception ex)
            {
                var msg = new ContentDialog();
                msg.Title = "エラーメッセージ";
                msg.Content = ex.Message;
                await msg.ShowAsync();
            }
        }
```

```
        //Firebaseへの送信処理
        private static async Task sendRun(FirebaseAuthLink link, string data)
        {
            try
            {
                var db = new FirebaseClient(
                       databaseURL,
                       new FirebaseOptions
                       {
                           AuthTokenAsyncFactory = () => Task.FromResult(link.FirebaseToken)
                       })
                       .Child("test/data");


                // データを格納する
                await db.PostAsync(new DatabaseData { Value = data, });
            }
            catch (Exception ex)
            {
                var msg = new ContentDialog();
                msg.Title = "エラーメッセージ";
                msg.Content = ex.Message;
                await msg.ShowAsync();
            }
        }
    }

    public class DatabaseData
    {

        public string Value { get; set; }

    }
}
```

ApiKey と databaseURL はプロジェクトに紐づけられておりブラウザで管理ツールにアクセスし取得する。２つの値は管理ツールのメイン画面から「ウェブアプリに Firebase を追加」をクリックし表示されるページに記載されている。



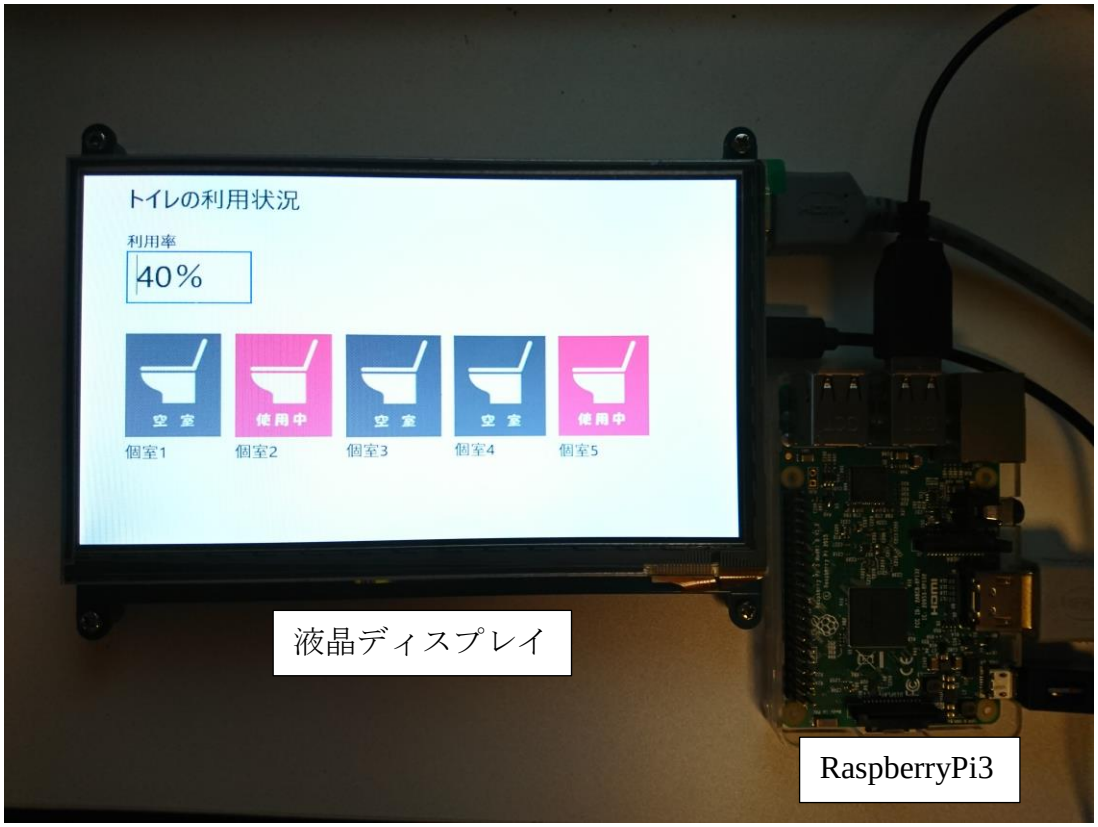
IoT ゲートウェイ層の実装

個室の利用状況可視化システムに利用する IoT ゲートウェイ層部分の実装を行う。センサデバイス層からの開閉状況を Bluetooth で受信し液晶ディスプレイに現在の個室の利用率を「%」表示し且つ個室 1 から個室 5 のそれぞれの利用状況を以下の画像を利用し表示する。

toiletempty.png

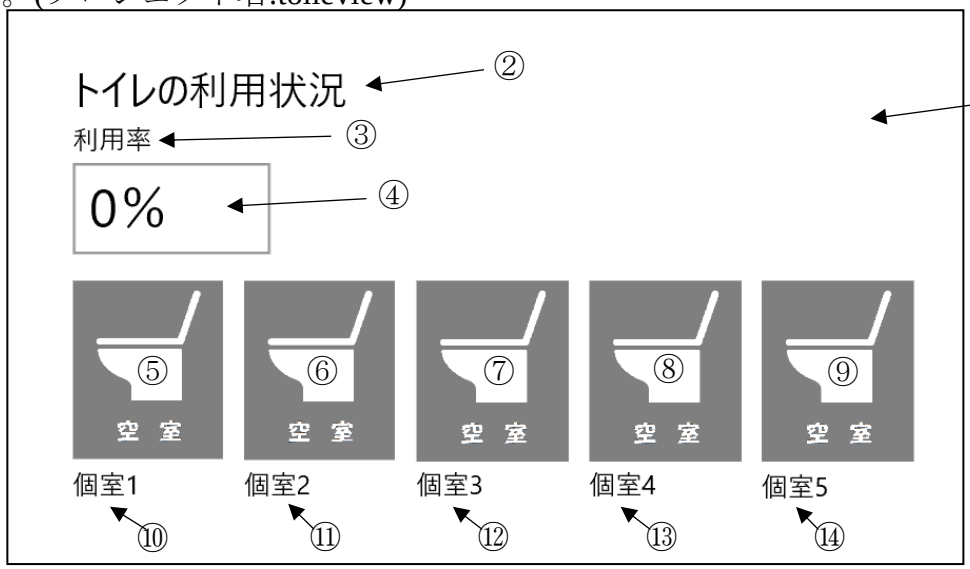
toiletuse.png

また、個室の利用率を Firebase Realtime Database へ保存する。



IoT ゲートウェイ層の実装課題

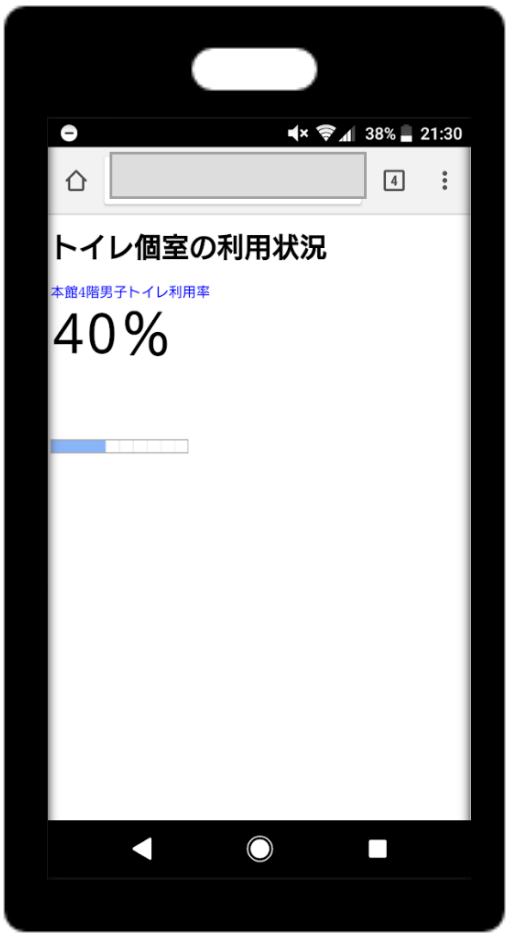
ＧＵＩは以下を参考とする。プロジェクトフォルダ内の Assets フォルダの中に空室,使用中の画像ファイルをそれぞれ「toiletempty.png」,「toiletuse.png」という名前で保存する。ここまで作成した Bluetooth 通信と Firebase との連携のアプリケーションを参考に IoT ゲートウェイ層を実装する。実装後は個室の開閉に合わせて液晶の表示が切り替わることと Firebase Realtime Database に利用率が格納されることを確認する。(プロジェクト名:toileview)



番号	コントロール種類	プロパティ項目名	プロパティ値
①	Grid	変更なし	変更なし
②	TextBlock	name	lblsystemname
		Text	トイレの利用状況
		FontSize	環境に合わせた適当な値
③	TextBlock	name	lblutilization
		Text	利用率
		FontSize	環境に合わせた適当な値
④	TextBox	name	txtutilization
		Text	0%
		FontSize	環境に合わせた適当な値
		IsReadOnly	true
⑤～⑨	Image	name	imgRoomNo1~imgRoomNo5
		Source	Assets/toiletempty.png
⑩～⑭	TextBlock	name	lblRoomNo1~lblRoomNo5
		Text	個室 1～個室 5
		FontSize	環境に合わせた適当な値

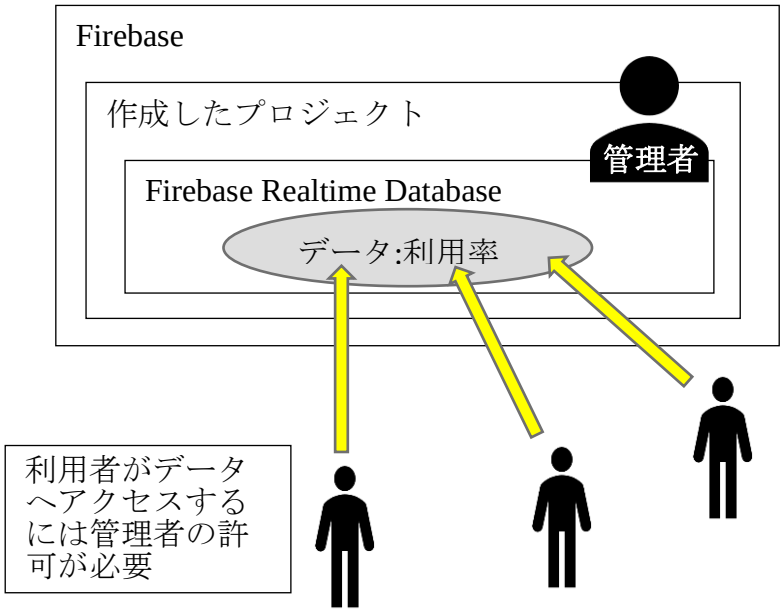
IoT クラウド層の構成

IoT クラウド層では、クラウドサービスを利用し個室の利用状況をリアルタイムに表示する Web サイトを提供する。IoT ゲートウェイ層において既に Firebase Realtime Database に利用率を保存するところまでは実装されている。この利用率を表示する為の Web サイトも Firebase のサービスを利用することで作成から公開までできる。



Firebase Realtime Database のルール設定

IoT ゲートウェイ層では、Firebase Realtime Database に現在の個室の利用率が保存されるように実装されている。Web サイト経由で利用者が保存された利用率を閲覧できるようにするためには Firebase Realtime Database の管理者がルールの設定を行う必要がある。



ルールは、細かな設定も可能だが今回はデータの読み取りだけは全ての人に対し許可を与えることとする。ルールを設定するにはブラウザから Firebase Realtime Database の管理ツールへアクセスする。「ルール」タブをクリックし表示されるエディタでルールの変更を行う。

Firebase Hosting の利用

Firebase Hosting は、Web サイトをホスティングすることのできるサービスである。Web サイト公開までの手順は公式サイトがドキュメントを参考となる。

Firebase Hosting の使用方法

https://firebase.google.com/docs/hosting/quickstart?hl=ja

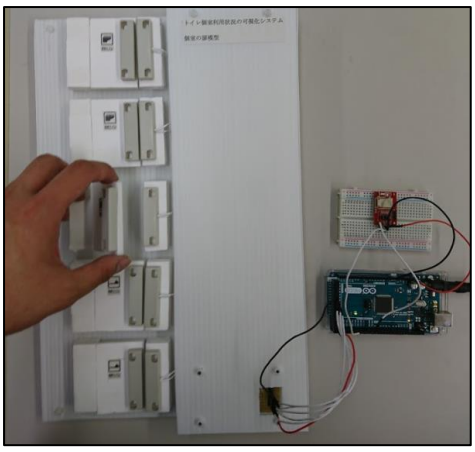
以下のコードをメモ帳等のエディターに入力しファイル名を index.html としデスクトップに保存する。コード中の太字斜体の部分は Firebase の管理コンソールにアクセスし必要なコードをコピーして貼り付ける。

index.html（サンプルコード）

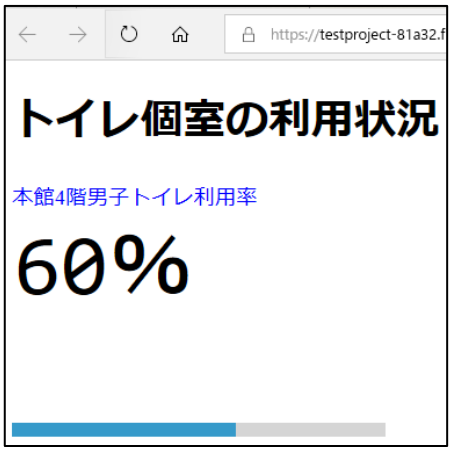
```
<!DOCTYPE html>
<html><head><meta charset="utf-8">
<title>トイレ個室利用状況</title>

<script src="https://www.gstatic.com/firebasejs/5.0.4/firebase.js"></script>
<script>
 // Initialize Firebase
 var config = {
  apiKey: "[redacted]"
  authDomain: "[redacted]",
  databaseURL: "[redacted]",
  projectId: "[redacted]",
  storageBucket: "[redacted]",
  messagingSenderId: "[redacted]"
 };
 firebase.initializeApp(config);
</script>

</head><body>
 <h1>トイレ個室の利用状況</h1>
 <font color="#0000FF">本館 4 階男子トイレ利用率</font>
 <div><textarea style="font-size:48pt; border:none;" name="text1" id="datavalue">
 </textarea></div>
 <div><progress max="100" id = "databar"><span>0</span>%</progress></div>
 <script>
 // Firebase のデータベースを取得
 var db = firebase.database();
 var data_db = db.ref("/test/data");
 // データが追加されたときの処理を記述
```

```
  function addLog(data) {
    var val = data.val();
    var e1 = document.getElementById("datavalue");
    var e2 = document.getElementById("databar");
    e1.value = "";
    e1.value = val.Value;
    //progressbar を更新
    e2.value = val.Value.replace("%","");
    e2.getElementsByTagName('span')[0].textContent = val.Value.replace("%","");
  }

  // データを追加したら内容を通知するよう設定
  data_db.limitToLast(1).on("child_added", addLog);
  function post() {
    var txt = document.getElementById("txt");
    data_db.push({ Value: txt.value });
    txt.value = "";
  }
  </script>
</body></html>
```

デスクトップ上に保存した index.html をダブルクリックすると web ブラウザが起動し以下のような Web サイトが表示される。



利用状況に合わせてリアルタイムに利用率が変化することを確認する



作成した index.html を Firebase の Hosting サービスを利用し外部へ公開する。ホスティングの作業は WindowsPowerShell を利用する。以下の手続きでホスティングを実行する。

① 以下のサイトにアクセスし Node.js と npm をインストールする。Node.js をインストールすると npm もインストールされる。

https://nodejs.org/

② WindowsPowerShell を起動し以下のコマンドを実行する。

```
npm install -g firebase-tools
```

③ 次に以下のコマンドを実行する。

```
firebase login
```

「Allow Firebase to collect anonymous CLI usage and error reporting information?」との質問がある。情報収集に協力するかの質問なので任意に Y(Yes)/N(No)どちらかを入力する。web ブラウザが起動しログイン画面が表示されるので Firebase で使用しているアカウントでログインする。

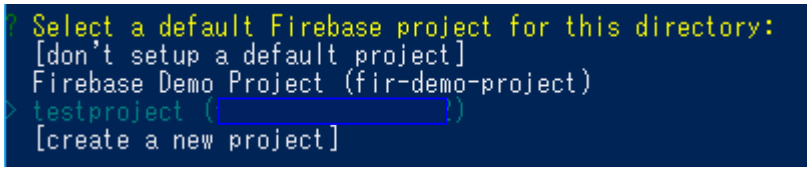
④ 次に以下のコマンドを実行する。

```
firebase init
```

「Are you ready to proceed?（続行していいか?）」の質問があるので Y と入力する。

⑤ 以下のような項目が表示されるのでカーソルキーで Hosting に合わせスペースキーで選択する。

⑥ 以下のようなプロジェクトの選択画面が表示されるので対象のプロジェクトを選択する。

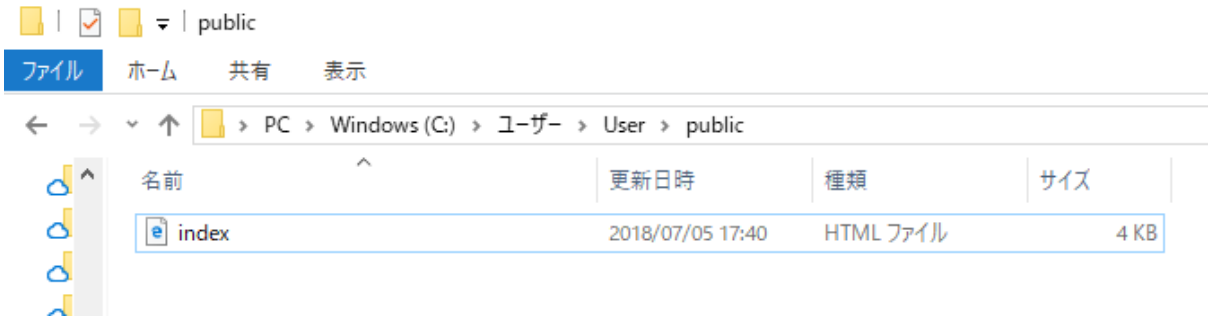


⑦ 以下の質問がされる。今回はデフォルトの状態を選択するのでそのまま Enter キーを入力する。これにより C ドライブ直下の各ユーザーフォルダ内に public というフォルダが生成される。

⑧ 続いて次の質問がされるが Y を入力する。

「Firebase initialization complete!」と表示される。

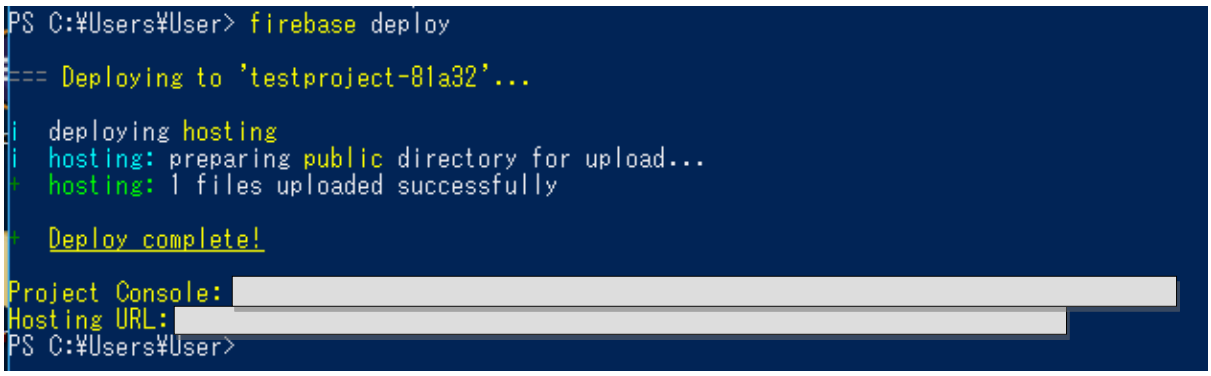
⑨ C ドライブ直下のユーザーフォルダ内に public というフォルダが生成されその中に index.html のファイルが生成されている。このファイルを先ほど作成した index.html のファイルに置き換える。



⑩ ファイルを置き換えたら PoweShell で以下のコマンドを実行する。

```
firebase deploy
```

⑪ 以下が表示され Hosthing URL の項目に書かれているアドレスにスマートフォン等の端末でアクセスする。



⑫ スマートフォン等の端末により公開された Web サイトを確認できる。もちろん利用状況に合わせてリアルタイムに利用率が変化することも確認でき作成した個室の利用状況をリアルタイムに確認できる Web サイトが外部に公開されたことが確認できる。

さいごに

本教材は実際に動作する IoT システムを構築しながら IoT を体感するのが目的であった。実習内で扱った技術は、ほんの一部で導入部に過ぎない。実際に IoT システムを扱うには高度な技術を必要とする。今日では IoT の普及に伴い IoT に特化した無線通信技術の存在やクラウド技術、IoT ならではのセキュリティの問題など学ばなくてはならない技術要素は多い。本教材が IoT とは何かを知るきっかけになることを期待している。

参考文献

[1]八子知礼・他『IoT の基本・仕組み・重要事項が全部わかる教科書』SB クリエイティブ社

[2]日本ユニシス社「IoT を構成する 3 階層」https://www.unisys.co.jp/solution/tec/iot/iot3layer.html

[3]KDDI 社「KDDI IoT クラウド ～トイレ空室管理～」https://iot.kddi.com/services/iot-cloud-toiletdoor/