IoT 技術教材

個室の利用状況の可視化システムの構築

秋田職業能力開発短期大学校

教材の活用方法(指導者用)

目次

教材の目的

本教材は、IoT を学ぶ際の導入として用いることを想定している。IoT を学びたい事情は様々で開発に携わっている人たちは IoT に必要な各要素のハイレベルな技術教育が必要となる。しかし、IoT の技術教育はそれだけでは無い。IoT が様々な分野で活用されていく中で、開発の間接部門にいる社員も「IoT とは何か」というのを当然のように知っておかなければいけない時代となっている。また、職業訓練をはじめとしたものづくり系の教育機関においても、IoT をテーマにしたカリキュラムが今後展開していくことも考えられる。既に IoT の言葉の意味など初歩的な内容を学ぶための教材は書店に並んでいる。本教材は、そこから更に技術的に一歩踏み込んで学べる内容となっており簡単なプログラミングを行いながら実際に動作する IoT システムを構築できる実習教材となっている。本教材のコンセプトは、「実際に動かしながら IoT の全体像を理解する」という事である。実習中に行うプログラミングも多いもので数十行程度と小規模にしてある。しかしその中でも IoT に必要な「無線通信技術」や「クラウド技術」なども取り入れてシステムを構築するので全体的なイメージをつかみやすい実習教材になっている。

教材の使用状況

本教材は、専門課程の総合制作実習で IoT をテーマに取り組む学生に対して IoT とは何かを理解させるために作成した教材で実際に学生に教材を適用した際は 4 単位程度（100 分×18）を要した。しかしこれは学生が自主的に取り組む総合制作実習であるという性質上学生自身が自分たちで考える時間を確保するために課題に多くの時間を費やしたり、授業では習わない技術要素を追加で学ぶための時間を含めてのものである。本教材はプログラミングの量も小規模であることから IoT をテーマとした在職者訓練にも適用できると考えている。参考までにと在職者訓練で教材を利用することを想定した場合の訓練カリキュラム案を次項で示す。

訓練カリキュラム

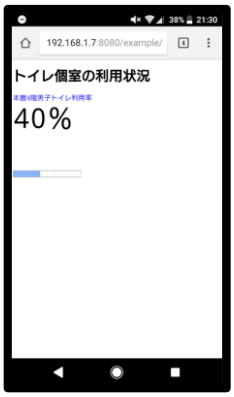
本教材を在職者訓練で利用する場合は、書籍等の教材だけではわからない IoT の全体像を実際に簡単なシステムを構築することで理解するのが大きな目的となる。実習で扱うプログラムは小規模なものなので何らかのプログラミング言語の経験者であれば十分に理解できる内容となっている。以下は在職者訓練で訓練時間 15 時間を想定した際のカリキュラム案である。

訓練対象者	IoTシステム構築に関心のある方 （マイコン等の未経験者も可、プログラミング言語によるプログラム作成の経験があればなお可）		
訓練目標	IoTの全体像を理解しIoTとは何かを実際に体験する。		
教科の細目	内容	訓練時間	実習
1.コース概要	(1)コース概要	1	
	(2)IoT基礎		
	(3)IoTシステムのアーキテクチャ		
	(4)無線技術		
	(5)クラウド技術		
2.システム構築実習	(1)「個室利用状況可視化システム」の概要	0.5	
	(2)センサデバイス層	5.5	4
	イ.Arduinoのプログラミング		
	ロ.Bluetoothモジュールを利用した通信プログラム		
	ハ.センサデバイス層の実装		
	(3)IoTゲートウェイ層	6	5.5
	イ. IoTゲートウェイの構成		
	ロ. IoTゲートウェイのプログラミング		
	ハ. Bluetooth通信プログラミング		
	ニ.クラウドとの連携プログラミング		
	ホ. IoTゲートウェイ層の実装		
	(4) IoTクラウド層	1.5	1
	イ．クラウドサービス		
	ロ. IoTクラウド層の実装		
3.まとめ	(1)実習の全体的な講評及び確認・評価	0.5	
		15	10.5

実習で作成するシステムの概要

教材で作成するIoTシステムは、個室の利用状況を可視化するシステムである。個室のドアの開閉状態を取得することで個室の利用状況を専用のWebサイトから確認できるサービスを提供する。実習では混雑するトイレの個室の利用状況を可視化することを題材として扱う。このようなシステムは既に小田急電鉄株式会社等が取り入れており身近なIoTシステムでもある。実習では、システムをデータ収集を行うセンサデバイス層、データ集約を行うIoTゲートウェイ層、データ活用を行うIoTクラウド層に分けて実装する。システムの構成は以下のようになる。

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

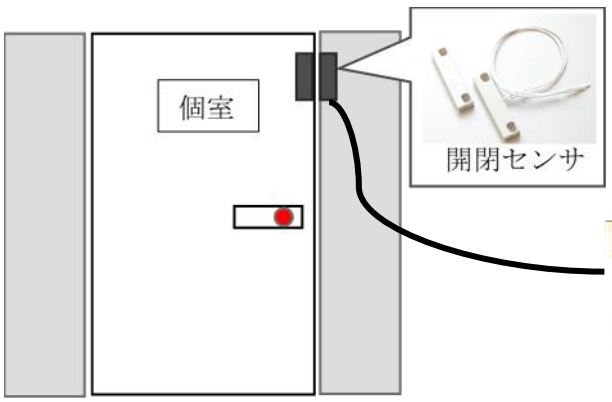
RaspberryPi3
オンボード上の通信モジュールを利用
個室利用状況の受信
・Bluetooth
クラウドへの利用状況送信
・Ethernet
・Wi-Fi

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

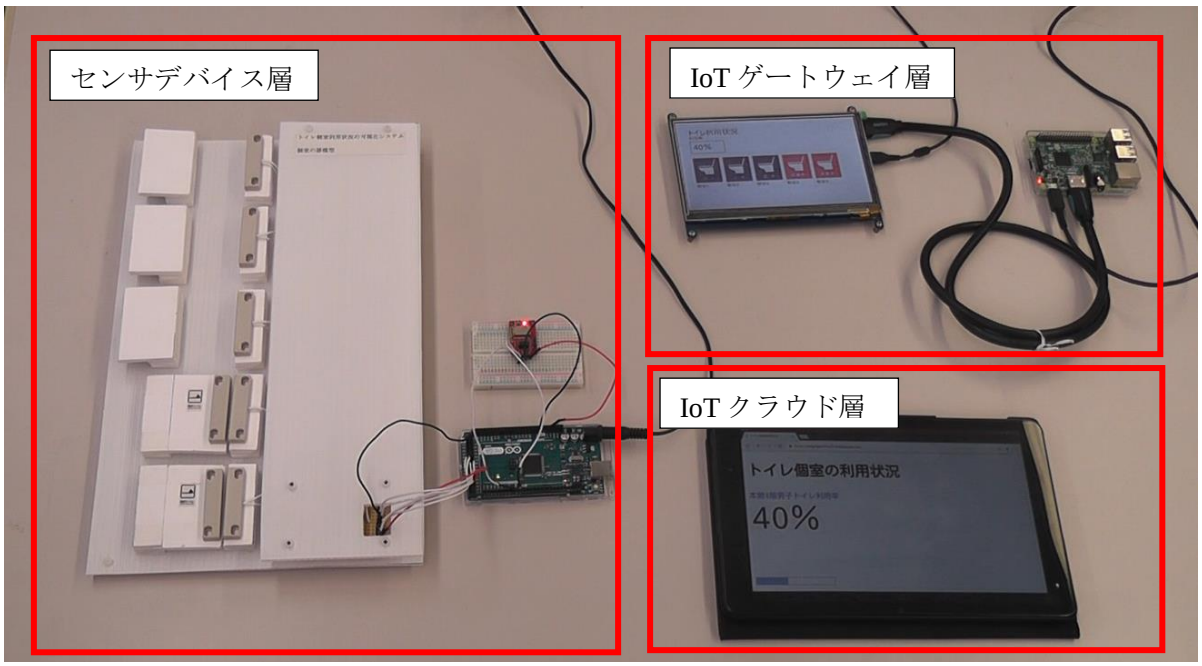
データ収集：センサデバイス層



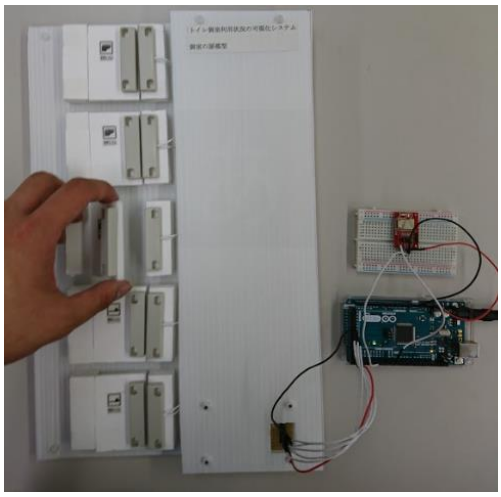
Bluetooth モジュール
通信範囲:30m 程度
通信速度:9600bps

Arduino
個室の利用状況の収集と
通信処理を実装

以下が実際の実習機材である。センサデバイス層は、トイレ個室の扉に取り付けた磁気リードスイッチから利用状況を取得し無線通信で、IoT ゲートウェイ層に送信する。IoT ゲートウェイ層では、液晶ディスプレイを通じて利用状況を通知するとともにクラウドに利用状況を送信する。IoT クラウド層では、クラウドサービスを利用し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。

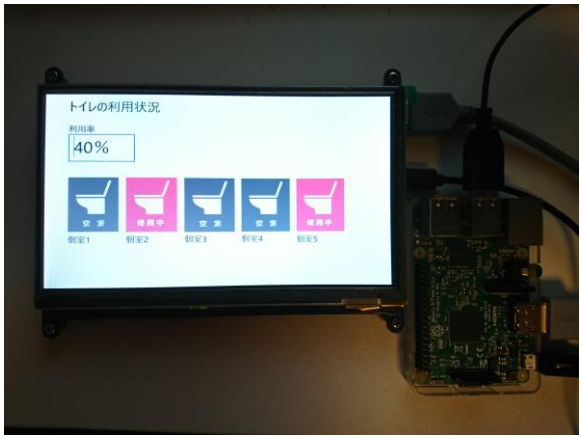


センサデバイス層の構成



- Arduino で処理を行う。
- 個室のドアを想定した入力装置を Arduino に接続しドアの開閉により利用状況を取得する。
- Bluetooth モジュールを利用し IoT ゲートウェイ層へ取得した利用状況を送信する。

IoT ゲートウェイ層の構成



- RaspberryPi3 に OS として Microsoft 社の Windows10IoT を搭載しアプリケーションを実行する。
- アプリケーションは、C#で作成しトイレ個室の利用状況をリアルタイムに液晶ディスプレイに表示する処理とクラウドに利用状況を送信する処理を実装する。

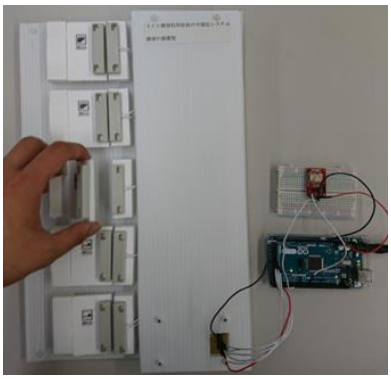
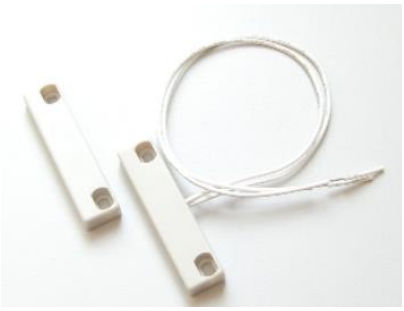
IoT クラウド層の構成

- クラウドサービスを利用しリアルタイムにトイレ個室の利用状況を確認できる Web サイトを提供する。
- Google 社の Firebase クラウドを利用し IoT ゲートウェイ層から送信されてきた利用状況の格納とそれを表示するための Web サイトのホスティングを行う

実習で作成するシステムは、電子データ CD-ROM に収録している「教材紹介.wmv」で確認できる。

実習を行う上で必要となる機材

ハードウェア

①	開発用パソコン	ユニバーサル Windows プラットホーム(UWP)アプリケーションを開発するため Windows10 が必須である。また開発者モードを有効にする必要がある。
②	Arduino	センサデバイス層の処理を行うプロセッサとして使用する。トイレ個室の利用状況取得と IoT ゲートウェイ層への利用状況の送信処理までを行う。実習テキストでは、例として Arduino Mega 2560 を利用している。
③	Raspberry Pi3	IoT ゲートウェイ層の処理を行うプロセッサとして使用する。Microsoft 社のエンベデットシステム用 OS である Windows 10 IoT Core を搭載し液晶ディスプレイに利用状況を表示する処理とクラウドへの利用状況の送信処理を行うアプリケーションを作成し動作させる。Raspberry Pi 3 は 5V 2.5A の電源が推奨される。
④	microSDHC カード	Windows 10 IoT Core のインストール用で 8GB 以上の容量を必要とする。
⑤	液晶ディスプレイ	Raspberry Pi で使用し HDML ケーブルで接続する。
⑥	マウス、キーボード	Raspberry Pi で使用する。
⑦	BLEserial3	浅草ギ研が販売する Bluetooth モジュールである。基板上のジャンパー設定で SSID を物理的に16種類に変更することができ、同時に複数のデバイスを使用できる。
⑧	個室を想定した入力装置	個室の利用状況はドアの開閉で判断する。ドアには開閉を検知するために磁気リードスイッチを取り付ける。  個室ドアを想定した装置  磁気リードスイッチ
⑨	ブレットボード、LED、トグルスイッチ、ジャンパー線	Arduino のプログラム実習や Bluetooth モジュールとの接続時に使用する。

ソフトウェア

①	Arduino IDE	Arduino の開発環境
②	Visual Studio	UWP アプリケーションを開発するため VisualStudio2015 以降を使用する。ユニバーサル Windows アプリ開発ツールをインストールする必要がある。

※Windows10IoTCore のインストール方法や設定方法、Windows10IoTCore で動作するアプリケーションの開発手順は別途本テキストの付録 A,B,C に示している。

指導案

システム構築実習の指導案を以下に示す。

指導項目	展開	時間(分)	教材
コース概要			
導入 動機	コースの概要を説明 ■ スケジュールの確認	5	
提示	訓練目標・目的の提示 ■ 実際に動作をするシステムを構築することで IoT とは何かを理解する。	5	
提示	IoT の基礎 ■ IoT とは ■ IoT と M2M についての説明 ■ IoT 実現に向けた各国の取り組み ■ IoT のアーキテクチャ例	20	パワーポイント資料 「IoT 基礎」 スライド番号:1 - 11
提示	IoT における通信技術 ■ Wi-Fi ■ Bluetooth ■ Zigbee ■ Z-Wave とメッシュ型ネットワーク ■ LPWA の特徴(LoRa 等)	20	パワーポイント資料 「IoT 基礎」 スライド番号:12 - 23
提示	IoT におけるクラウドサービス ■ クラウドサービスの概要	10	パワーポイント資料 「IoT 基礎」 スライド番号:24 - 25
トイレ個室利用状況可視化システムの概要			
導入 動機	実習で作成するシステム「個室の利用状況の可視化システム」の概要を説明 システムの目的を説明 ・作成するシステム例として小田急電鉄株式会社の「小田急アプリ」の例を示す ・実習で作成するシステムの動作を動画で見せる	10	動画ファイル 「教材紹介.wmv」
	IoT システムのアーキテクチャの説明 ■ IoT システムは仕組みや利用方法は様々である ■ アーキテクチャは仕組みによって変化する ■ 実習ではセンサデバイス層、IoT ゲートウェイ層,IoT クラウド層として表現する ■ 各層の役割	10	実習テキスト P2

	実習で作成するシステムのアーキテクチャ ■ 各層に分けてシステムの構成を説明 これから作成するシステムのイメージをつかめたかを確認	10	実習テキスト P4
休憩(10 分)			
センサデバイス層の実装			
展開 動機	センサデバイス層の実装への導入 ■ センサデバイス層の構成を説明	5	実習テキスト P6
提示	センサデバイス層の実装における目標 ■ Arduino の基本的なプログラミングができる ■ シリアル通信のプログラミングができる ■ Bluetooth モジュールの利用方法を習得する	5	
提示	Arduino 基板の概要とピン配置などの仕様を調べる手順を提示	5	実習テキスト P6-P9
適用	実習で使用する Arduino 基板の仕様とピン配置を調べてもらう。	10	
提示	Arduino の開発環境について説明する	10	実習テキスト P10
提示	Arduino のプログラミングで利用する関数について説明 ■ 関数の調べ方	15	実習テキスト P11-P14
適用	Arduino に LED を接続し LED を点灯させるプログラムを作成（作業画面を提示しながら一緒に作業する） ■ LEDの接続ピンを確認 ■ 使用する関数の説明 ■ プログラミングと動作確認	25	実習テキスト P15-P17
提示	ここまでの作業のポイントを提示	5	
休憩(60 分)			
提示	Arduino プログラミング課題1の説明	5	実習テキスト P18
適用	Arduino プログラミング課題1を実施 ■ 様子を見ながら使用する関数を提示	15	
提示	Arduino プログラミング課題1の解説	5	
提示	Arduino プログラミング課題 2 の説明	5	実習テキスト P18

適用	Arduino プログラミング課題 2 を実施 ■ 様子を見ながら配列の解説や for 文の解説	30	
提示	Arduino プログラミング課題 2 の解説	10	
提示	Arduino プログラミング課題 3 の説明	5	実習テキスト P19
適用	Arduino プログラミング課題 3 を実施 ■ 様子を見ながら入力処理について解説	25	
提示	Arduino プログラミング課題 3 の解説	10	
休憩(10 分)			
提示	Arduino におけるシリアル通信の概要 ■ ハードウェアシリアルとソフトウェアシリアルについて解説 ■ シリアル通信のピンについて解説	5	実習テキスト P19
適用	シリアル通信のプログラムを作成し動作確認を行う	10	実習テキスト P20
提示	Arduino プログラミング課題 4 の説明 ■ Arduino 言語における String 型の説明を行う	5	実習テキスト P21
適用	Arduino プログラミング課題 4 を実施	15	
提示	Arduino プログラミング課題 4 の解説	5	
提示	Arduino プログラミング課題 5 の説明	5	実習テキスト P21
適用	Arduino プログラミング課題 5 を実施 ■ 様子を見ながらヒントを与える	45	
提示	Arduino プログラミング課題 5 の解説	10	
提示	実習で利用する Bluetooth モジュールの説明 ■ モジュールと Arduino の接続を説明	10	実習テキスト P22
適用	センサデバイス層の実装 ■ 課題 5 を流用することで実装できることを説明 ■ データを送信する際は改行コードを送信しないようにする。(以降のサンプルプログラムが動作しなくなる)	15	実習テキスト P23
提示	ここまでの作業のポイントを提示	5	
IoT ゲートウェイ層の実装			
展開 動機	IoT ゲートウェイ層の実装への導入 ■ IoT ゲートウェイ層の構成を説明	10	実習テキスト P24
提示	IoT ゲートウェイ層の実装における目標 ■ Windows10IoTCore の概要を理解する ■ C#でアプリケーションを作成する ■ サンプルコードをもとに Bluetooth 通信のプログラムが作成できる ■ サンプルコードをもとにクラウドとの連携プログラムを作成できる	5	

提示	IoT ゲートウェイの構成を説明 ■ RaspberryPi3 について ■ エンベデットシステム用の OS である Windows10IoTCore の概要について ■ RaspberryPi3 上で動作する Windows10IoTCore の各種設定（使用言語、ネットワーク設定、コンピュータ名、ディスプレイ設定） ※設定は講師側で事前設定	15	実習テキスト P24-P25
1日目終了			
展開 提示	GUI アプリケーションの作成①（作業画面を提示しながら一緒に作業する） ■ 開発環境 VisualStudio の使い方 ■ デザインエディターとコードエディター ■ デザインエディターでの作業 ■ コードエディターでの作業 ■ イベントの登録 ■ クラスとオブジェクト ■ メソッドとプロパティ ■ アプリケーションの配置手順 ■ 動作確認	45	実習テキスト P26-P29
展開 提示	GUI アプリケーションの作成② ■ 構造化例外処理について解説	45	実習テキスト P30-P33
休憩(10 分)			
展開 提示	Bluetooth 通信を実行するアプリケーション ■ Bluetooth 通信をアプリケーションで利用する場合のマニフェストファイルの変更について ■ サンプルコードの提示 ■ アプリケーションの作成作業 ■ 動作確認	60	実習テキスト P34-P37
提示	クラウドサービスについて ■ クラウドサービスの概要 ■ Backend as a service について ■ 実習で利用するクラウドサービス Firebase について	10	実習テキスト P38-P39
展開 提示	Firebase の利用方法（作業画面を提示しながら一緒に作業する） ■ Google アカウントでのログイン ■ プロジェクトの作成 ■ 利用するサービス Firebase Realtime Database と Hosting について	20	実習テキスト P40-P41
休憩(60 分)			
展開 提示	クラウドとの連携アプリケーションの作成（作業画面を提示しながら一緒に作業する） ■ サンプルコードの提示 ■ アプリケーションの作成作業	60	実習テキスト P42-P48

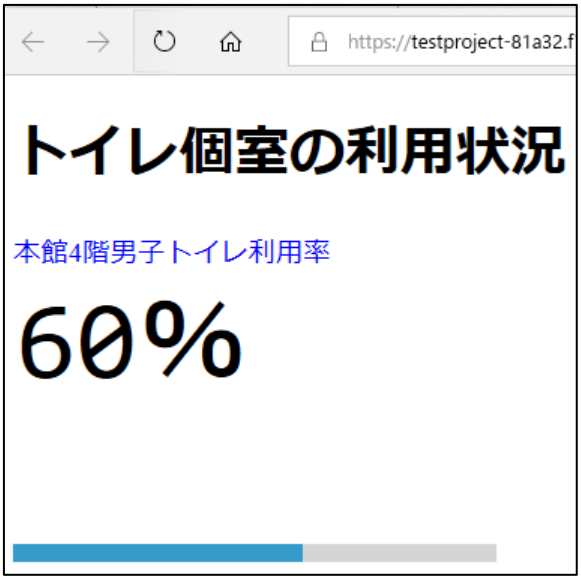
	■ 動作確認		
展開	IoT ゲートウェイ層の実装 ■ ここまで作成してきたアプリケーションを流用組み合わせて実装できることを説明	90	実習テキスト P49-P50
提示	ここまでの作業のポイントを提示	10	
休憩(10 分)			
IoT クラウド層の実装			
展開 動機	IoT クラウド層の実装への導入 ■ IoT クラウド層の構成を説明	5	実習テキスト P51
提示	IoT ゲートウェイ層の実装における目標 ■ クラウドのサービスについて理解する	5	
展開 提示	Firebase Hosting サービスを利用する際の手順を提示し利用するための作業を行う(作業画面を提示しながら一緒に作業する)	15	実習テキスト P51-P56
展開	利用状況を確認することができる Web サイト用の html ファイルを作成する。(作業画面を提示しながら一緒に作業する) ■ 作成したWebサイトを Hosting サービスを利用し公開する ■ スマートフォン等の端末を利用し作成した Web サイトにアクセスし動作を確認する	45	実習テキスト P51-P56
展開	トイレ個室利用状況可視化システムとして動作していることを最終確認する	15	
提示	ここまでの作業のポイントを提示	5	
まとめ			
展開	実習のまとめ ■ 質疑応答 ■ 理解度の確認	30	
2日目終了			

テキスト利用時の解説ポイント

ページ	解説ポイント
P2	IoT システムのアーキテクチャについて説明する。 ① IoT システムのアーキテクチャは仕組みにより異なる。 ② 教材ではデータを収集の役割を担う「センサデバイス層」、データを集約する役割を担う「IoT ゲートウェイ層」、データの活用を担う「IoT クラウド層」の 3 つの層でシステムを構築し説明、実装を行う。
P3-P5	実際に作成する個室の利用状況可視化システムを P2 で説明したアーキテクチャに照らし合わせながら説明する。 ① 個室の利用状況はドアの開閉で検知する。その際は磁気リードスイッチを利用する。利用状況はデータを集約している IoT ゲートウェイ層のマイコンボードに無線通信で送信する。 ② センサデバイス層から送られてきた利用状況はリアルタイムに利用者に通知するためまずは液晶ディスプレイに利用状況を表示する。それと同時にクラウドに利用状況を送信する。 ③ IoT クラウド層では、利用状況をトイレ利用者がスマートフォン等の端末で確認できるように専用の Web サイトを提供する。利用状況の格納等はクラウドサービスを活用する。
P6	センサデバイス層の構成を説明する。 ① 利用状況の取得と IoT ゲートウェイ層への利用状況の送信までを実装する。 ② 処理を行うマイコン基板は Arduino を利用する。
P6-P9	Arduino 基板について説明する。 ① Arduino の概要 ② 製品ラインナップ ③ 使用する Arduino 基板の仕様の調べ方
P10-P17	Arduino のプログラミングの基礎を説明する。 ① 開発環境について ② 関数の調べ方 ③ LED 点灯のプログラム作成をする。LED は正論理接続とする。 ④ プログラムの書き込み
P18-P19	Arduino のプログラミング課題を行う ① 課題 1 は、関数の調べ方を習得してもらう。 ② 課題 2 は、配列の使い方を習得してもらう。 ③ 課題 3 は、デジタル入力処理を習得してもらう。
P19-P20	Arduino のシリアル通信プログラムを説明する。 ① ハードウェアシリアルとソフトウェアシリアルについて ② サンプルコードを入力する ③ 開発環境に付属するシリアルモニタを利用し動作確認を行う。
P21	Arduino のプログラミング課題を行う ① 課題 4 は、Arduino における String 型を理解してもらう。 ② 課題 5 は過去のデータと新しいデータを比較し変化があった時だけ送信する。
P22	Bluetooth モジュールを説明する。

	① 実習で使用する Bluetooth モジュールは浅草ギ研が販売しているもので Arduino の RX,TX ピンとモジュールの RX,TX ピンを適切に接続するだけでペアリングした機器にデータを送受信できる。 ② Arduino 側のプログラムは単純なシリアル通信のプログラムで実装できる。
P23	センサデバイス層を実装する。 ① トイレ個室を想定した入力装置と Bluetooth モジュールを Arduino に接続する。 ② プログラムは Arduino プログラミング課題 5 を流用可能である。ただしデータを送信する際は改行コードは送信しないように注意する。以降のサンプルプログラムは改行コードが送信されないことを前提に作成している。
P24	IoT ゲートウェイ層の構成について説明する。 ① センサデバイス層からの利用状況を Bluetooth で受信し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処理を行うとともにクラウドに利用状況を送信するところまでを実装する。 ② 処理は RaspberryPi3 で行い Microsoft のエンベデットシステム用 OS の Windows10IoTCore を搭載する。 ③ RaspberryP3 はオンボード上に Bluetooth と Wi-Fi のモジュールが搭載されているためセンサデバイス層からの利用状況の受信とクラウドへの利用状況の送信はこれらを使用する。
P25	Windows10IoTCore の概要を説明する。 ① Microsoft のエンベデットシステム用 OS である。IoTCore はライセンス料無料で使用できる。 ② OS は Microsoft の公式ドキュメントに従って簡単にインストールできる。時間がかかるので講師側であらかじめ SD カードにインストールしておく。(付録 A 参照) ③ 開発環境 VisualStudio で C#などでアプリケーションを開発できる。
P26- P27	アプリケーション開発について説明する。 ① VisualStudio の使い方 ② デザインエディターで GUI 作成をする。講師が画面を提示しながら一緒に作業する。 ③ イベントの登録の方法を説明する。講師が画面を提示しながら一緒に作業する。 ④ コードエディターでコードを入力する。講師は画面を提示しながらコード入力の際は intellisense 機能の活用について説明する。 ⑤ 講師はリモートでのデバックを画面に提示しながら一緒に作業する。(付録 C 参照)
P28- P29	C#でプログラミングする際の最低限必要になる知識を説明する。 ① クラス、プロパティ、メソッド、イベント等について最低限必要の知識について解説する。 ② 必要があれば専門書等を併用して指導する。
P30- P33	同様にアプリケーション開発について説明する。 ① 先ほどと同様に講師は画面を提示しながら一緒に作業する。 ② ここでは例外処理について解説する。これも必要があれば専門書等を併用して指導する。

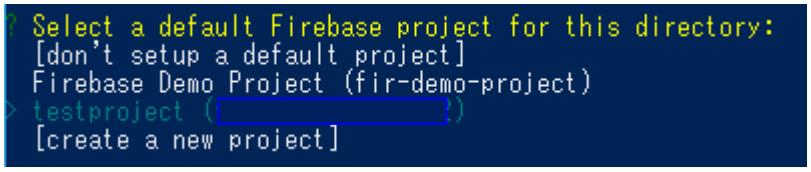
P34- P37	Bluetooth 通信のアプリケーションを作成する。 ① Bluetooth 通信を実現するためのクラスがある。 ③ Microsoft の公式ドキュメントに掲載されているサンプルをもとに作成したプログラムを提示する。 ④ Bluetooth をアプリケーションで使用するためのマニフェストファイルの変更を説明する。講師は画面を提示しながら一緒に作業する。 ⑤ コードを入力しアプリケーション配置後動作を確認する。
P38- P48	クラウドとの連携アプリケーションを作成する。 ① クラウドは Google の Firebase を利用する。Firebase は無料枠内で使用する。 ② Google アカウントを準備する。Firebase は準備した Google アカウントで使用する。通常使用している Google アカウントでもよい。 ③ 講師は画面を提示しながら Firebase にログインし手順に従ってプロジェクトを作成する。 ④ Firebase で使用する Database と Hosting のサービスを解説する。 ⑤ まず Database サービスを利用しクラウドにデータを格納するアプリケーションを作成する。 ⑥ クラウドにデータを格納するために C#用の SDK を利用することを説明する。 ⑦ 講師は画面を提示しながらプロジェクトへの C#用の SDK の追加からコード入力までを一緒に作業する。 ⑧ アプリケーションを配置し動作確認を行う。
P49- P50	IoT ゲートウェイ層を実装する。 ① 利用状況を液晶ディスプレイに表示する処理とクラウドへ利用状況を利用率として格納するところまでを実装する。 ② 液晶ディスプレイに表示される画面の構成を指定する。 ③ 使用中、空室は画像の切り替えにより表現する。その画像は事前に配布する。 ④ 使用中、空室の画像をプロジェクトの Assets フォルダにコピーし VisualStudio のソリューションエクスプローラからプロジェクトに追加する。この時講師は画面を提示しながら一緒に作業する。 ⑤ GUI を作成しアプリケーションの実装を行う。受講生には今まで作成した Bluetooth の通信、クラウドとの連携アプリケーションは流用してできることを説明する。 ⑥ 状況に合わせて使用中、空室の画像切り替えの部分と利用率を計算して表示するまでの処理をヒントとして提示する。 ⑦ 動作確認として利用状況に合わせて液晶ディスプレイ上の画面が切り替わることとクラウドへ利用率が格納されることを確認する。
P51	IoT クラウド層の構成を説明する。 ① IoT ゲートウェイ層の実装により個室の利用状況が利用率としてクラウド上に格納されている。その利用状況をリアルタイムに確認できる Web サイトを提供する。 ② 利用状況を確認できる Web サイトを提供するために HTML と Javascript で Web サイトを制作する。Firebase のサービスを利用するための Web 用の SDK が提供されているのでそれを利用し作成することを説明する。
P52	Database サービスのルール設定について解説する。

	① クラウド上に格納されている利用状況を Web サイトで確認できるようにするには、その利用状況は誰でも読み取れるようにしなくてはいけない。そのためのルールを設定する必要がある。
P53- P54	サンプルコードをもとに HTML と Javascript でコードを入力する。 ① メモ帳等のエディタでコードを入力し index.html でデスクトップ上に保存する。Firebase のサービスを利用するために Web 用の SDK を利用していることを説明する。 ② 保存した index.html をダブルクリックするとブラウザが起動し以下のような Web サイトが表示されることを確認する。利用状況によって利用率が変化することを確認する。 

P55-
P57
	クラウドの Hosting サービスを利用し外部に Web サイトを公開する。
	① Hosting の作業は WindowsPowerShell へのコマンド入力で実行する。講師が画面を提示しながら一緒に作業する。
	② 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/
	③ WindowsPowerShell を起動し以下のコマンドを実行する。
	`npm install -g firebase-tools`
	④ 次に以下のコマンドを実行する。「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。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。
	`firebase login`
	⑤ 次に以下のコマンドを実行する。「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。
	`firebase init`

⑥ 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。

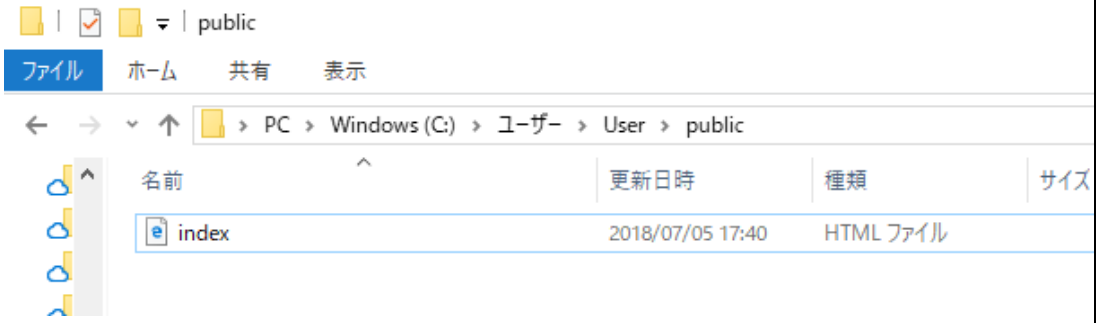
⑦ 以下のようなプロジェクトの選択画面が表示されるので対象のプロジェクトを選択する。



⑧ 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 ドライブ直下のユーザーフォルダの中のユーザー名フォルダ内に public というフォルダが生成される。ここに先ほどの index.html のファイルを置くことになる。

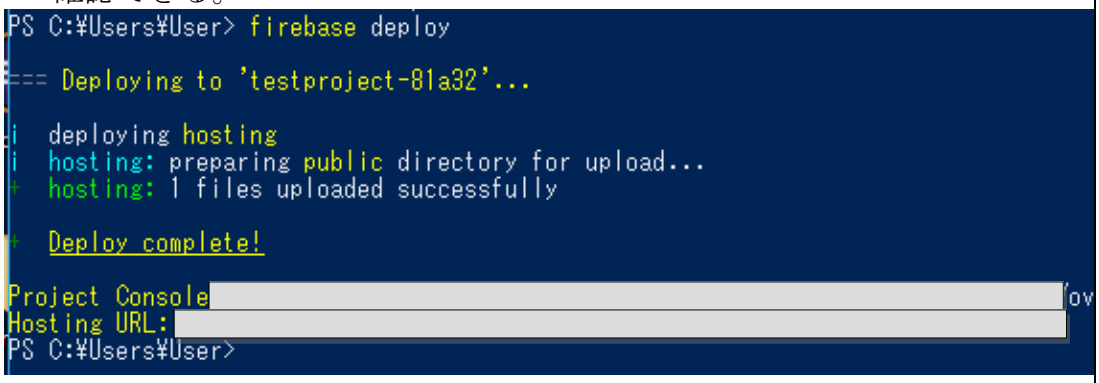
⑨ 続いて次の質問がされるが Y を入力する。「Firebase initialization complete!」と表示される。

⑩ C ドライブ直下のユーザーフォルダの中のユーザー名フォルダ内に 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る。このファイルを先ほど作成したファイルに置き換える。



⑪ 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
```

	⑫ 以下が表示され Hosthing URL の項目に書かれているアドレスに受講生のスマートフォン等の端末でアクセスさせる。公開された Web サイトが確認できる。 

実習課題サンプルプログラム

Arduino プログラミング課題 1 サンプルコード

サンプルコード arduinokadai1

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

void setup() {
  // put your setup code here, to run once:
  //30 番ピンを出力モードに設定
  pinMode(outpin1,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  //low レベルの信号を出力
  digitalWrite(outpin1, LOW);
  //500ms 待ち
  delay(500);
  //High レベルの信号を出力
  digitalWrite(outpin1, HIGH);
  //500ms 待ち
  delay(500);
}
```

Arduino プログラミング課題2 サンプルコード

サンプルコード arduinokadai2

```
//30-34 番ピンを利用
const int outpins[] = {30, 31, 32, 33, 34};


//led 点灯パターン
const int leddata[] = {
  0b00000001,
  0b00000010,
  0b00000100,
  0b00001000,
  0b00010000
};

void setup() {
  // put your setup code here, to run once:
  //30-34 番ピンを出力モードに設定
  for(int init = 0;init<5;init++)
  {
    pinMode(outpins[init],OUTPUT);
  }

}

void loop() {
  // put your main code here, to run repeatedly:
  for(int datano = 0;datano<5;datano++)
  {
    printled(datano);
    delay(500);
  }
}

void printled(int n){
    for (int i = 0; i < 5; i++) {
    //条件式 ? 真の場合の文 : 偽の場合の文
    digitalWrite(outpins[i], leddata[n] & (1 << i)? HIGH : LOW);
  }
}
```

Arduino プログラミング課題3 サンプルコード

サンプルコード arduinokadai3

```
//30-34 番ピンを利用
const int outpins[] = {30, 31, 32, 33, 34};
//40-44 番ピンを利用
const int inpins[] = {40, 41, 42, 43, 44};

//led 点灯パターン
int leddata[] = {0,0,0,0,0};

void setup() {
  // put your setup code here, to run once:
  //30-34 番ピンを出力モードに設定
  for(int init = 0;init<5;init++)
  {
    pinMode(outpins[init],OUTPUT);
  }

}

void loop() {
  // put your main code here, to run repeatedly:
  for(int count = 0;count<5;count++)
  {
    leddata[count] =digitalRead(inpins[count]);
  }
  printled(leddata);
}

void printled(int n[]){
    for (int i = 0; i < 5; i++) {
    //条件式 ? 真の場合の文 : 偽の場合の文
    digitalWrite(outpins[i], n[i] ? HIGH : LOW);
  }
}
```

Arduino プログラミング課題 4 サンプルコード

サンプルコード arduinokadai4

```
//40-44 番ピンを利用
const int inpins[] = {40, 41, 42, 43, 44};

//sw のデータ
int data[] = {0,0,0,0,0};
//送信データ
String strdata="";

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  for(int count = 0;count<5;count++)
  {
    data[count] =digitalRead(inpins[count]);
  }
  serialprint(data);
  delay(1000);
}

void serialprint(int n[]){
    for (int i = 0; i < 5; i++) {
    strdata += String(n[i]);
  }
  Serial.println(strdata);
  strdata="";
}
```

Arduino プログラミング課題 5 サンプルコード

サンプルコード arduinokadai5

```
//40-44 番ピンを利用
const int inpins[] = {40, 41, 42, 43, 44};

//sw のデータ
int data[] = {0,0,0,0,0};
//古いデータ
String olddata="00000";
//新しいデータ
String newdata="00000";

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  for(int count = 0;count<5;count++)
  {
    data[count] =digitalRead(inpins[count]);
  }
  serialprint(data);
  delay(500);
}

void serialprint(int n[]){
    for (int i = 0; i < 5; i++) {
    newdata += String(n[i]);
  }
  //古いデータと新しいデータを比較し違ったら実行
  if(!olddata.equals(newdata))
  {
    //古いデータを更新
    for (int j = 0; j < 5; j++) {
      olddata[j] = newdata[j];
    }
    //データ送信
    Serial.println(newdata);
  }
  newdata = "";
}
```

IoT ゲートウェイ層の実装

サンプルコード プロジェクト名:toileview

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



using Windows.Devices.Bluetooth.GenericAttributeProfile;
using Windows.Storage.Streams;
using Windows.Devices.Bluetooth;
using Windows.UI.Xaml.Media.Imaging;
using Windows.Devices.Enumeration;

using Firebase.Auth;
using Firebase.Database;
using System.Threading.Tasks;
using Firebase.Database.Query;
using System.Threading;

// 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 toileview
{
    /// <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
    {


        //使用中,空き画像
        private BitmapImage bitmapempty = new BitmapImage(new Uri("ms-appx:///Assets/toiletempty.png"));
        private BitmapImage bitmapuse = new BitmapImage(new Uri("ms-appx:///Assets/toiletuse.png"));
        private Image[] arrayImage = new Image[5];
        //private List<Image> arrayImage = new List<Image>();

        //Bluetooth
        private Guid SERVICE_UUID = new Guid("FEED0001-C497-4476-A7ED-727DE7648AB1");
        private Guid CHARACTERSTI_UUID = new Guid("FEEDAA03-C497-4476-A7ED-727DE7648AB1");
        private DeviceInformationCollection deviceInfos;
        private GattCharacteristicsResult gattCharacteristics;

        private static string ApiKey = "AIzaSyBlparuzaga59wnEzfcGmKTkFVoeyorl-g";
        public const string databaseURL = "https://testproject-81a32.firebaseio.com/";
        private static string AuthEmail = "test@example.com";
        private static string AuthPassword = "password";



        private FirebaseAuthLink authLink;

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

```
        private void Grid_Loaded(object sender, Windows.UI.Xaml.RoutedEventArgs e)
        {

            this.arrayImage[0] = imgRoomNo1;
            this.arrayImage[1] = imgRoomNo2;
            this.arrayImage[2] = imgRoomNo3;
            this.arrayImage[3] = imgRoomNo4;
            this.arrayImage[4] = imgRoomNo5;

            this.BluetoothAsync();
            this.firebaseconnect();

        }

        //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();
                        // CCCD への書き込みが必要。これがないとイベントハンドラが呼ばれない。
                        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();
            }
        }

        //Bluetooth受信処理
        private async void GattCharacteristicChange(GattCharacteristic sender, GattValueChangedEventArgs args)
        {

            byte sum = 0;
```

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


            await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, async () =>
            {
                try
                {
                    for (int i = 0; i < bArray.Length; i++)
                    {
                        if (bArray[i] == '1')
                        {
                            arrayImage[i].Source = bitmapuse;
                            sum += 1;
                        }
                        else
                        {
                            arrayImage[i].Source = bitmapempty;
                        }
                    }
                    txtutilization.Text = (((double)sum / 5) * 100).ToString() + "%";
                    await sendRun(authLink,(((double)sum / 5) * 100).ToString() + "%");
                }
                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, });
            }
```

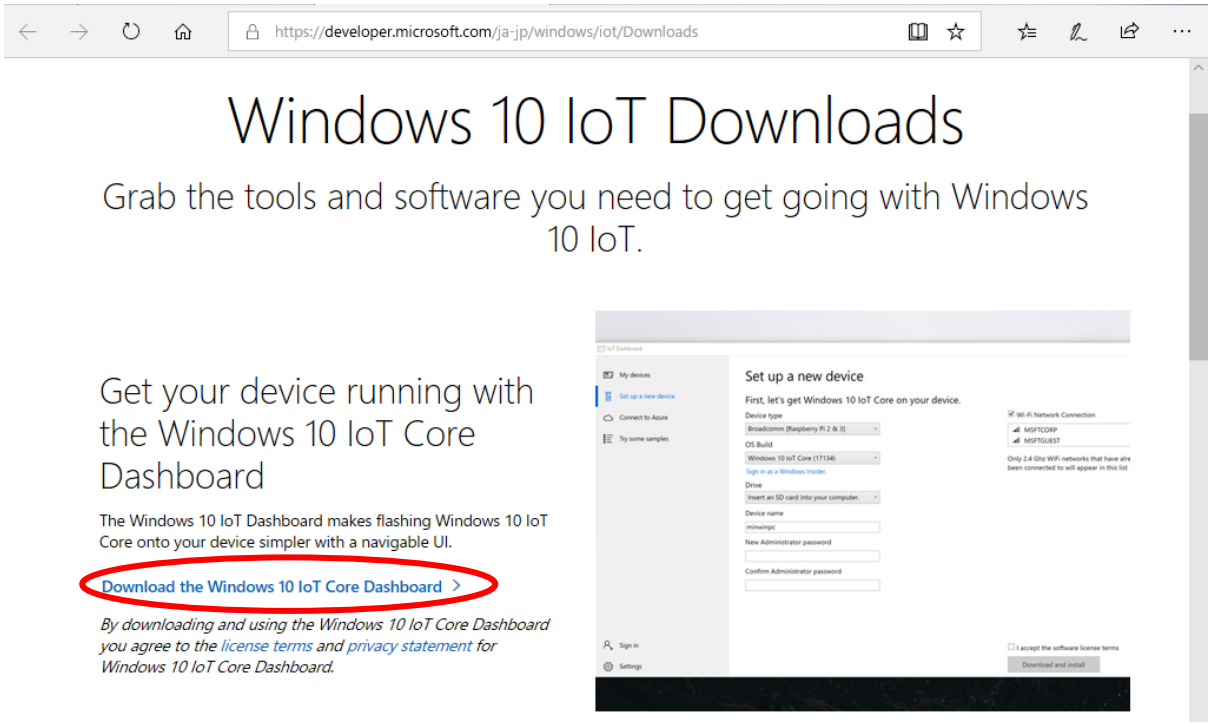
```
            catch(Exception ex)
            {
                var msg = new ContentDialog();
                msg.Title = "エラーメッセージ";
                msg.Content = ex.Message;
                await msg.ShowAsync();
            }
        }
    }

    public class DatabaseData
    {

        public string Value { get; set; }

    }
}
```

付録 A:Windows10IoTCore のインストール手順

RaspberryPi3 で Windows10IoTCore を利用するには SD カードに OS のイメージを書き込む必要がある。SD カードは容量 8G バイト以上の microSD カードで Class10 以上のものが推奨される。

① インストールを行うには Microsoft のホームページにアクセスし「Windows 10 IoT Core Dashboard」をダウンロードする。

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



② ダウンロードが完了すると Windows 10 IoT Core Dashboard が起動する。

③ フォーマット済みの SD カードを Windows 10 IoT Core Dashboard をダウンロードしたパソコンにセットし、IoT Dashboard の画面からデバイスの種類を「Broadcomm[Raspberry Pi2 & 3]、OS ビルドを「Windows 10 IoT Core」、ドライブをセットした SD カードを選択しデバイス名と管理者パスワードを入力してから「ダウンロードとインストール」ボタンをクリックする。この操作により SD カードに OS イメージを書き込むことができる。

④ OS イメージの書き込みが終了したら SD カードを取り出し RaspberryPi にセットする。RaspberryPi に液晶ディスプレイを接続し起動すると以下のような画面が表示される。これが OS の起動画面である。

※OS のバージョンにより表示が異なる場合がある。

付録 B:Windows10IoT の各種設定

ネットワーク設定の手順

① 設定を行うために RaspberryPi にマウスとキーボードを接続する。RaspberryPi に電源を投入しネットワーク設定を行う。

② ネットワークに接続するには Eternet での接続、Wi-Fi での接続どちらでも良い。実習環境に合わせて選択する。

【Ethernet 接続】

LAN ケーブルを接続する。

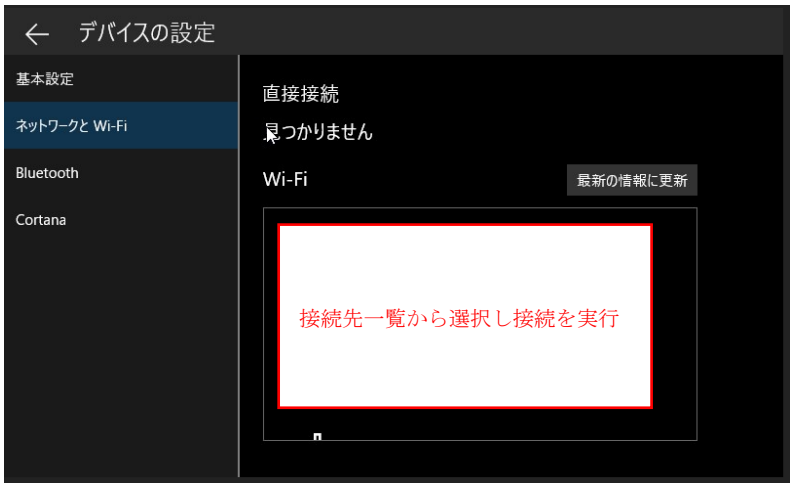
DHCP の環境下では接続するだけで IP アドレスが振られるので確認する。

【Wi-Fi 接続】

設定（歯車アイコン）をクリックする

ネットワークと Wi-Fi タブをクリックする

接続できる Wi-Fi の一覧が表示されるので接続先を選択し接続を実行する。



トップの画面に戻り割り振られた IP アドレスを確認する。

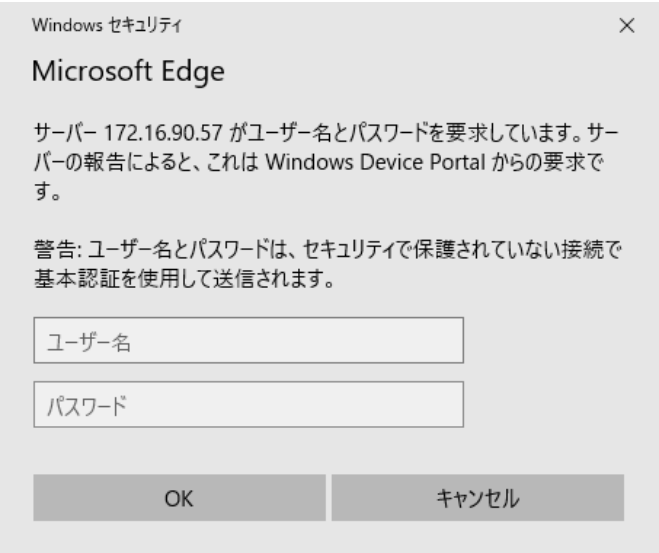
各種設定

IP アドレスが割り振られたら Web ブラウザ経由でアクセスし各種設定が可能となる。

Web ブラウザを起動しアドレス欄に以下のアドレスを入力する。

http://割り振られた IP アドレス:8080/

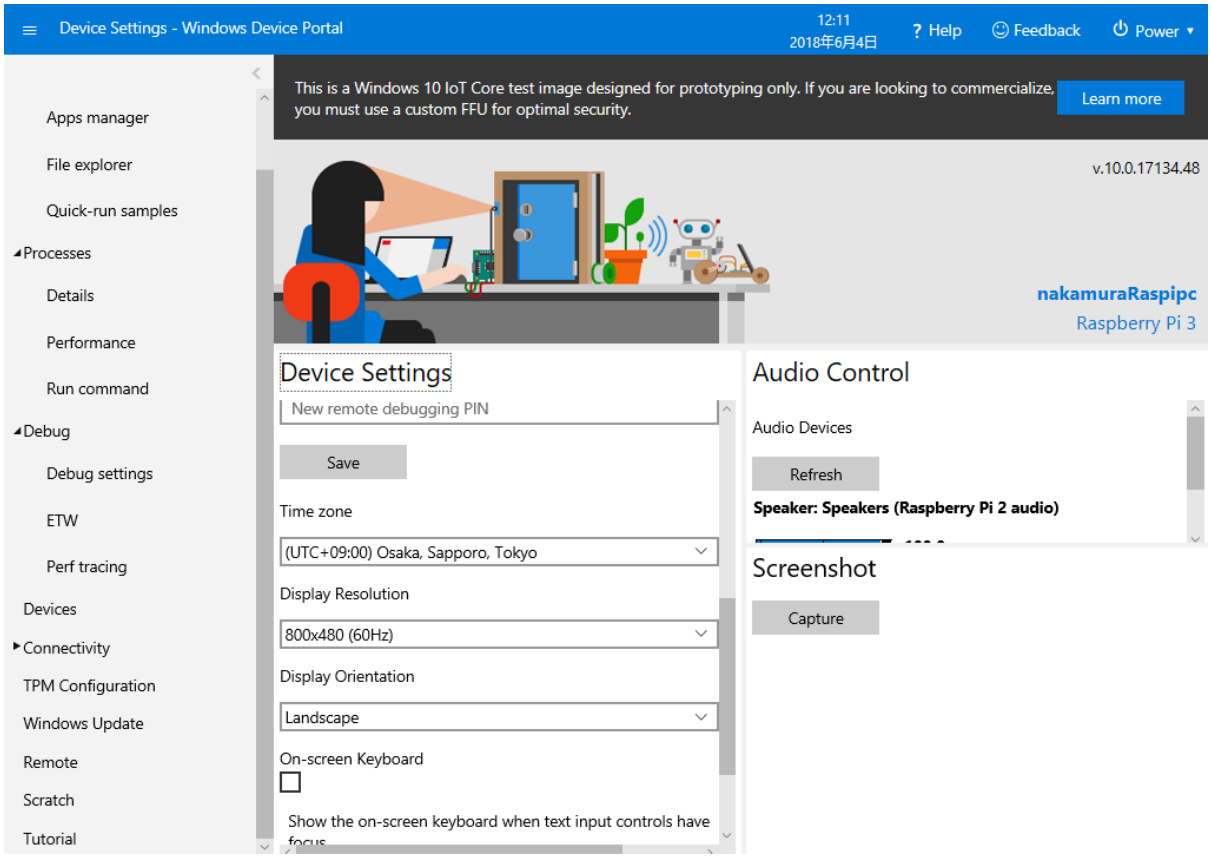
以下のような認証画面が表示されるのでユーザー名とパスワードを入力し「OK」をクリックする。



ユーザー名：administrator

パスワード：OS イメージをインストールするときに設定したパスワード

認証画面からすると管理画面が表示される。この管理画面から各種設定を行うことができる。最低限液晶ディスプレイの解像度設定は使用する環境に合わせて設定の必要がある。



Bluetooth の有効化

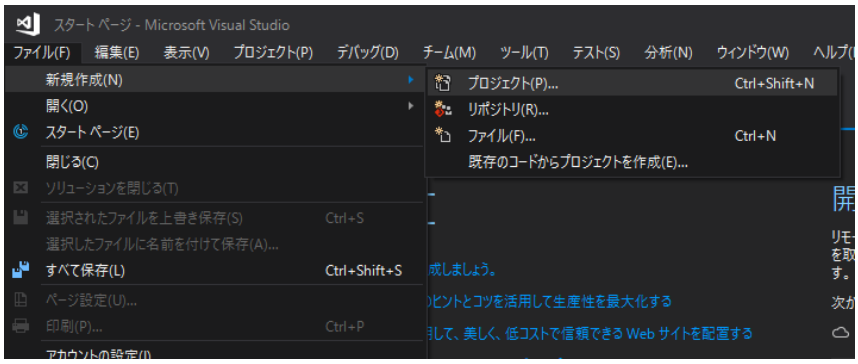
実習では Bluetooth 通信を利用するため設定で Bluetooth を有効にする必要がある。設定（歯車アイコン）をクリックする。

Bluetooth を ON にする。ここではペアリング等の操作は必要ないので設定を ON にするだけで良い。

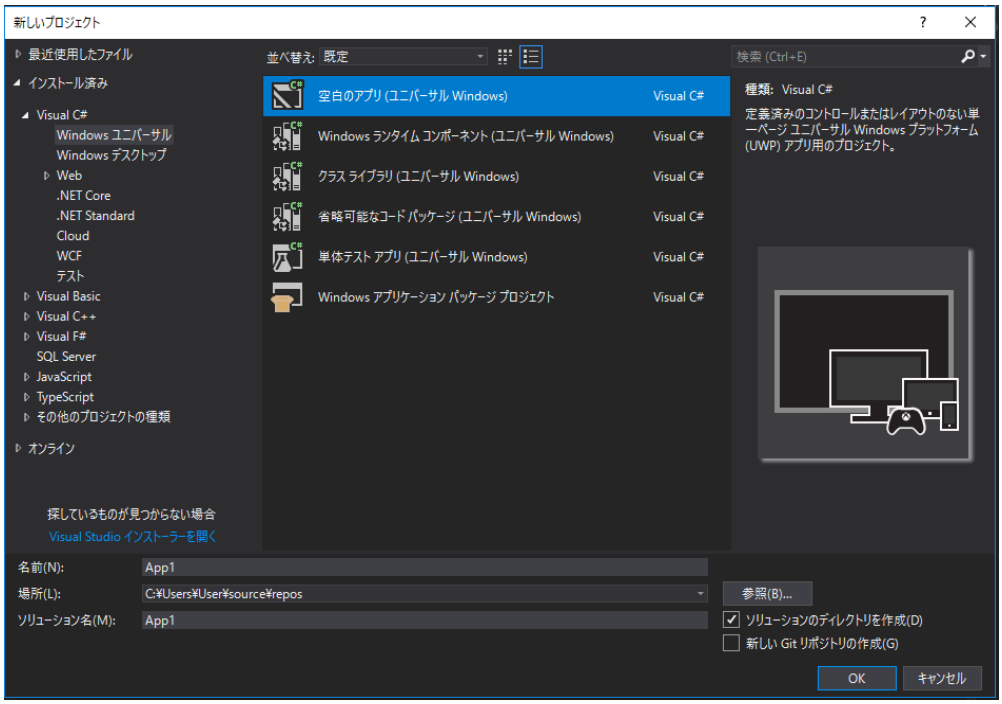
付録 C:Windows10IoT で動作するアプリケーションの開発手順

Windows10IoT 上で動作するアプリケーションは VisualStudio を利用して作成する。

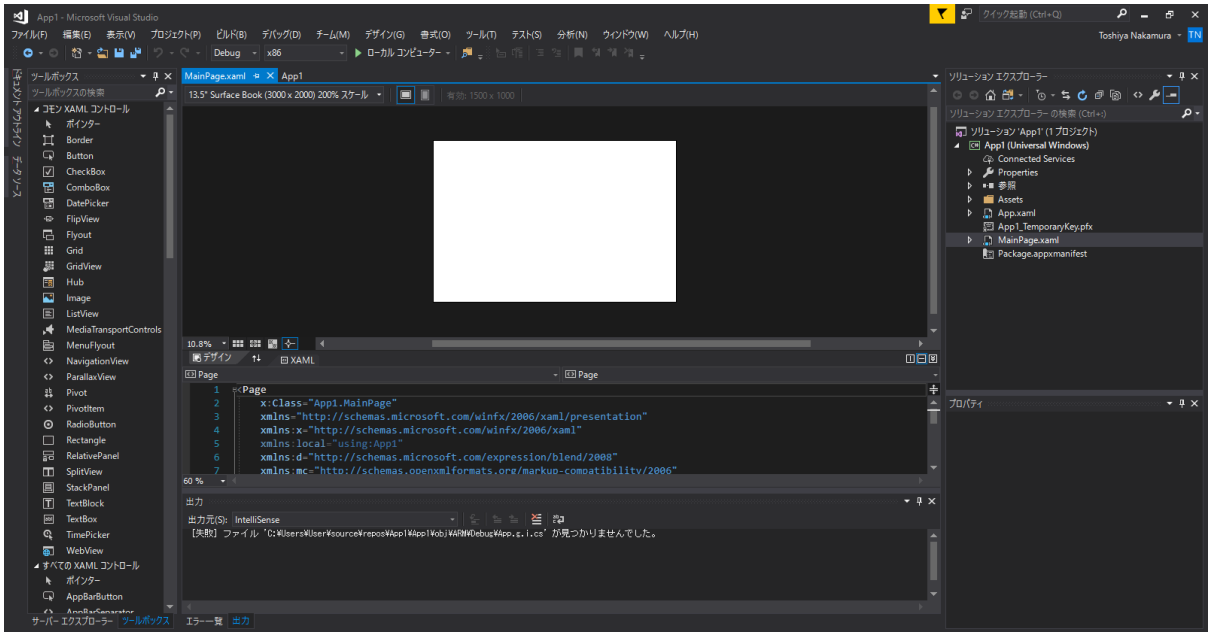
① VisualStudio を起動しメニューバーの「ファイル」から「新規作成」、「プロジェクト」を選択する。



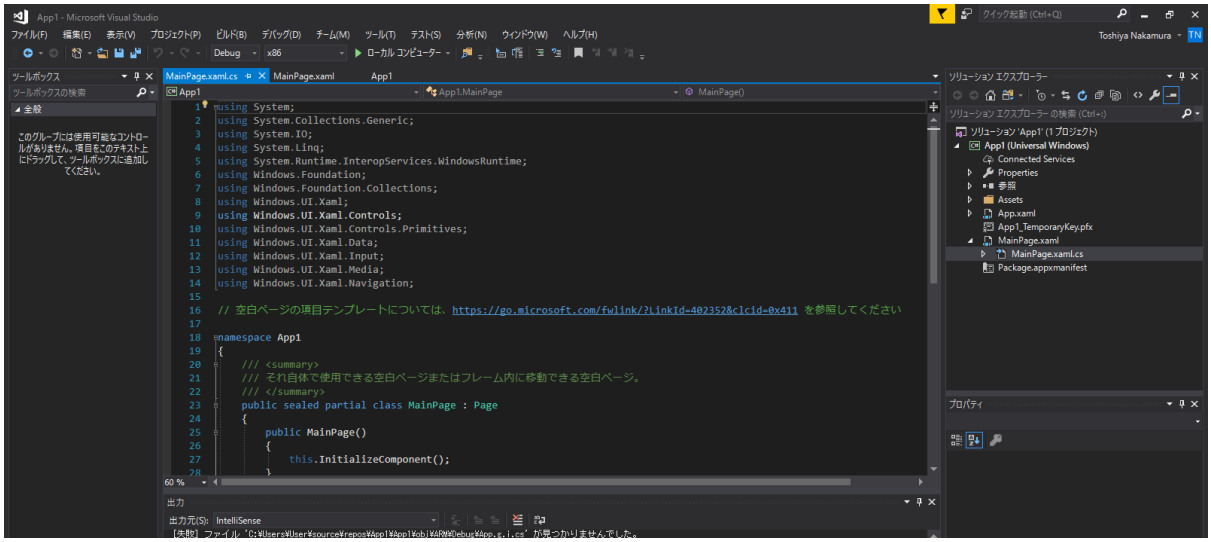
② 新しいプロジェクトウィンドウが起動するのでアプリケーションのテンプレートとして「Visual C#」、「Windows ユニバーサル」、「空白のアプリ」を選択しプロジェクトの保存先プロジェクト名を入力し「OK」をクリックする。Windows ユニバーサルの開発は、開発ツールをインストールする必要がある。



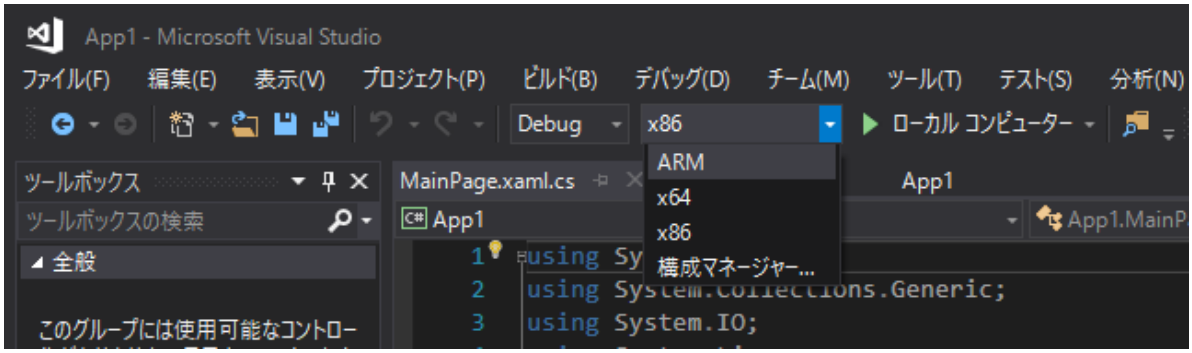
③ アプリケーションの作成自体は通常のデスクトップアプリケーションの作成するときと大きな違いは無い。ソリューションエクスプローラから「MainPage.xaml」をダブルクリックすることでデザインエディターが起動する。



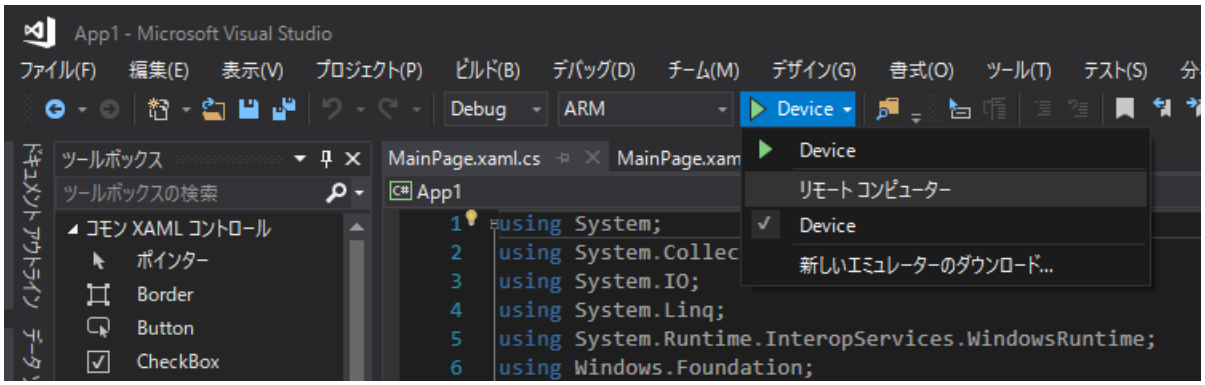
④ デザインエディターから「MainPage.xaml.cs」をダブルクリックすることでコードエディターが起動する。



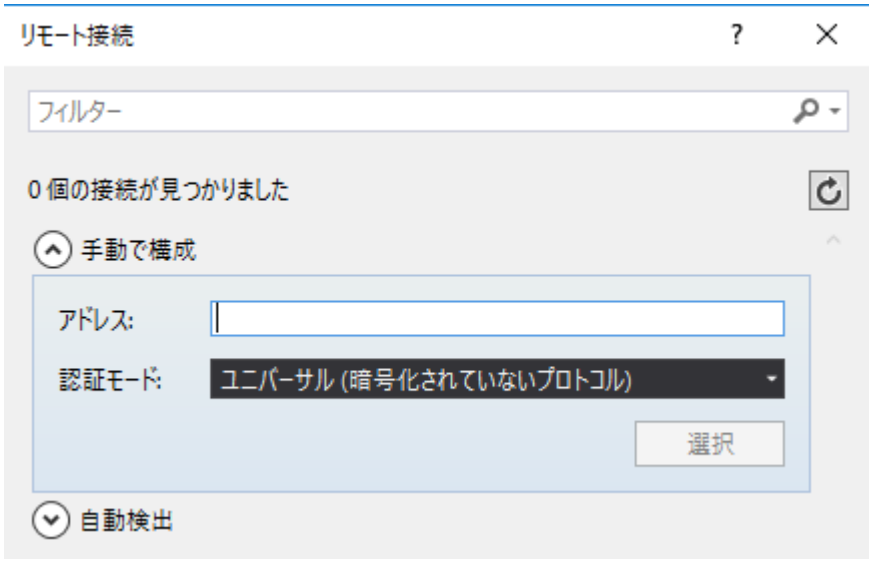
⑤ アプリケーション作成後は開発を行っているパソコンからリモートでデバックを行う。今回は RaspberryPi を利用しているのでビルドターゲットを ARM に変更する。



⑥ ツールバーの緑色の三角形ボタンの右側のドロップダウンボタンをクリックする。一覧からリモートコンピュータを選択する。



⑦ リモート接続のダイアログが表示されるのでアドレス欄に RaspberryPi に割り振られている IP アドレスを入力し「選択」をクリックする。



⑧ ツールバーの緑色の三角形ボタンをクリックしてデバッグを開始する。RaspberryPi に作成したアプリケーションが配置され起動する。