KC4-C-100A

LTE™ Category 4搭載 多機能ゲートウェイ

ブロックプログラミング※をはじめよう

通信プロトコル編①

Ver.1.0

※ブロックを構成して作るビジュアルプログラミングを
ブロックプログラミングと定義しています。

通信プロトコル編①について

本ドキュメントは「ブロックプログラミングをはじめよう」の「導入編」の続編となります。このためレシピツールのインストールやレシピの機器への書込み等は記載されていないため、最初に「導入編」をご参照ください。

- はじめるまでの準備
- キッティングツールについて
- レシピツールについて
- ブロックプログラミングについて
- はじめてのブロックプログラミング

通信プロトコル編①について

本編は以下の通信プロトコルの使い方を説明致します。

- HTTP
- MQTT

※通信プロトコル編②にて以下を説明致します。

- HTTPS
- MQTTS

なお、本編ではSIMとクラウドサービスはSORACOMを使用して説明しているため、SORACOM IoT SIMをご準備いただきますと理解が深まります。

通信プロトコルブロックを使用する準備

本章では通信プロトコル関係のブロックを使用する前の準備を説明します。

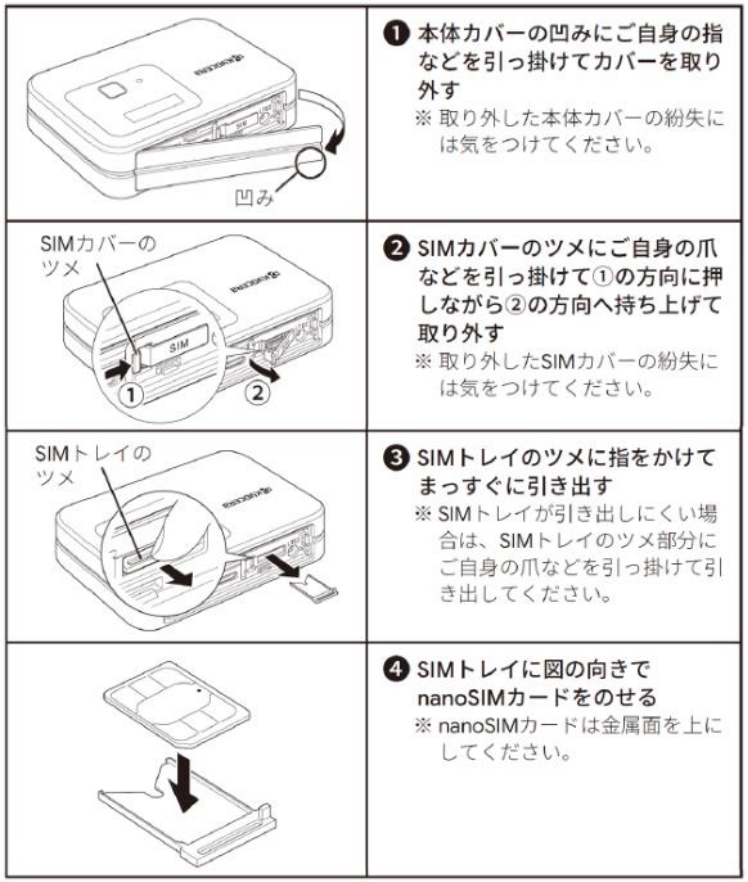
- SIMの挿入
- キッティングツールの操作
- APN/認証パラメータの設定

を行います。

SIMの挿入

本製品に電源を入れる前にSIMを挿入します。

1.SIMの入れ方

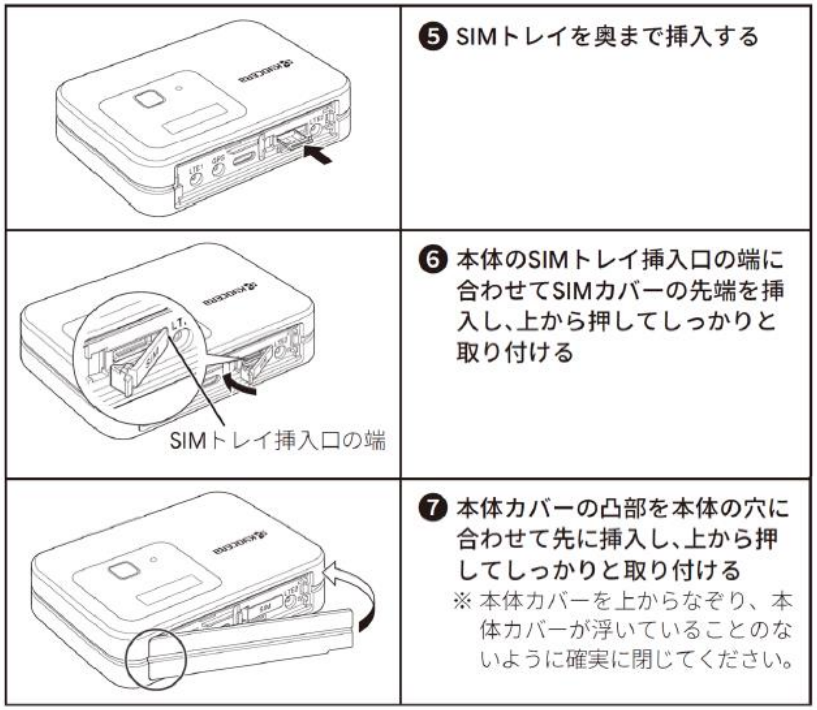


❶ 本体カバーの凹みにご自身の指などを引っ掛けてカバーを取り外す
※ 取り外した本体カバーの紛失には気をつけてください。

❷ SIMカバーのツメにご自身の爪などを引っ掛けて①の方向に押しながら②の方向へ持ち上げて取り外す
※ 取り外したSIMカバーの紛失には気をつけてください。

❸ SIMトレイのツメに指をかけてまっすぐに引き出す
※ SIMトレイが引き出しにくい場合は、SIMトレイのツメ部分にご自身の爪などを引っ掛けて引き出してください。

❹ SIMトレイに図の向きでnanoSIMカードをのせる
※ nanoSIMカードは金属面を上にしてください。



❺ SIMトレイを奥まで挿入する

❻ 本体のSIMトレイ挿入口の端に合わせてSIMカバーの先端を挿入し、上から押してしっかりと取り付ける

❼ 本体カバーの凸部を本体の穴に合わせて先に挿入し、上から押してしっかりと取り付ける
※ 本体カバーを上からなぞり、本体カバーが浮いていることのないように確実に閉じてください。

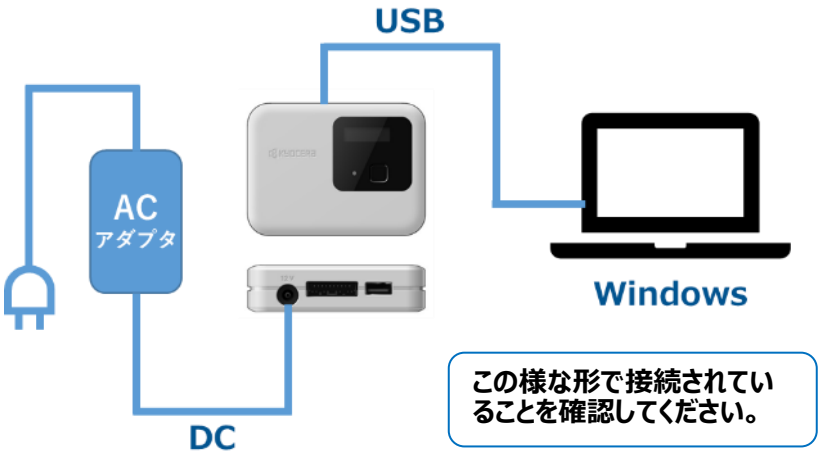
※機器の扱い方は合わせて同梱の取り扱い説明書をご参照願います。

キッティングツールの操作

挿入したSIMの接続先を設定するために、キッティングツールを操作します。

1.キッティングツールの立上げ

キッティングツールの立上げの前に、PCに機器を下記のように接続してください。



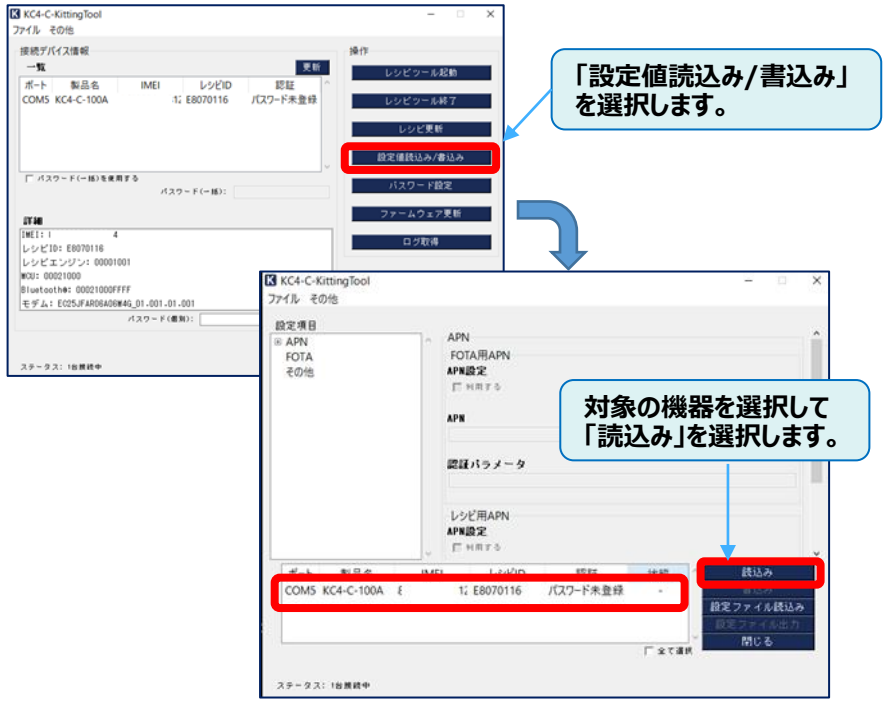
この様な形で接続されていることを確認してください。

レシピツールが正しくインストール完了しているとデスクトップに「K」アイコンが表示されるので、このアプリケーションを実行します。

 ← このアイコン

2.APNの設定①

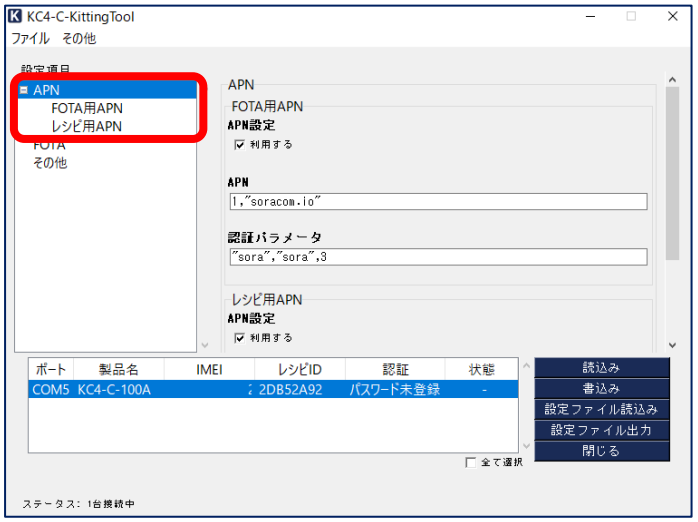
キッティングツールが立ち上がると以下のような画面が表示されます

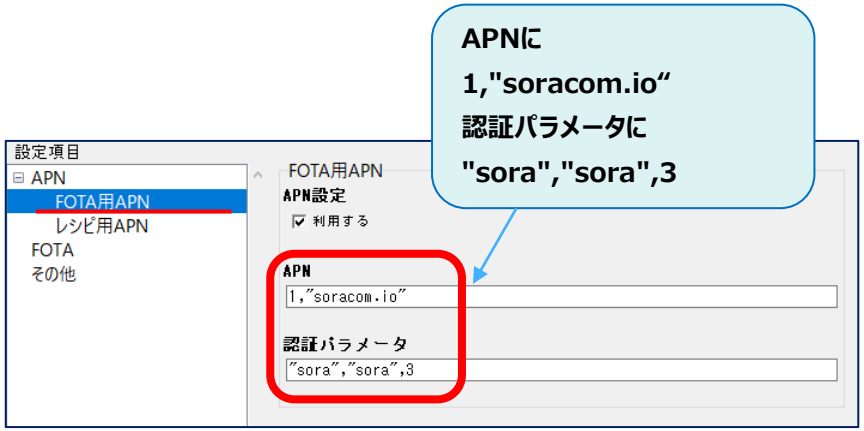


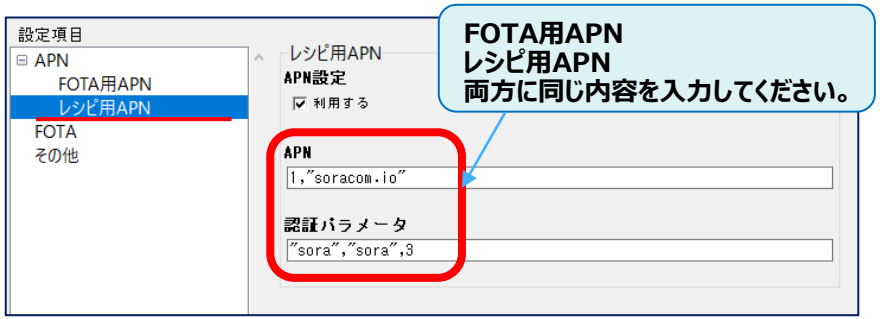
キッティングツールの操作

3.APNの設定②

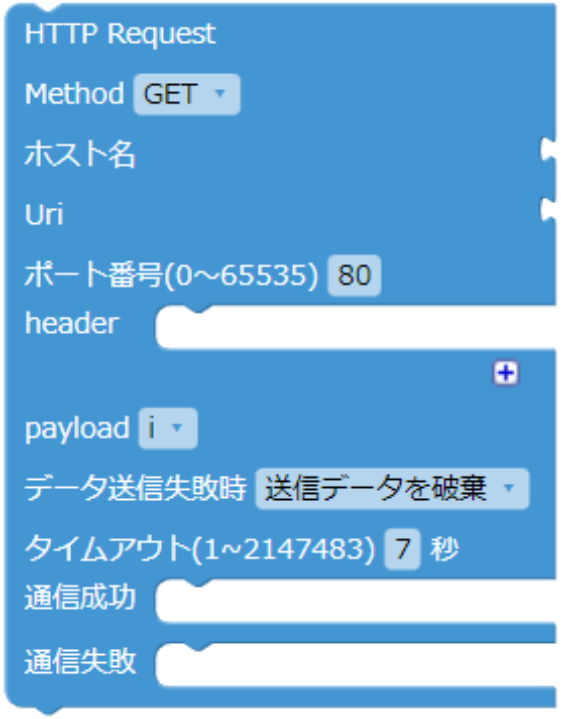
APNのメニューを選択して、APNの設定を行います。







HTTPブロックの使い方

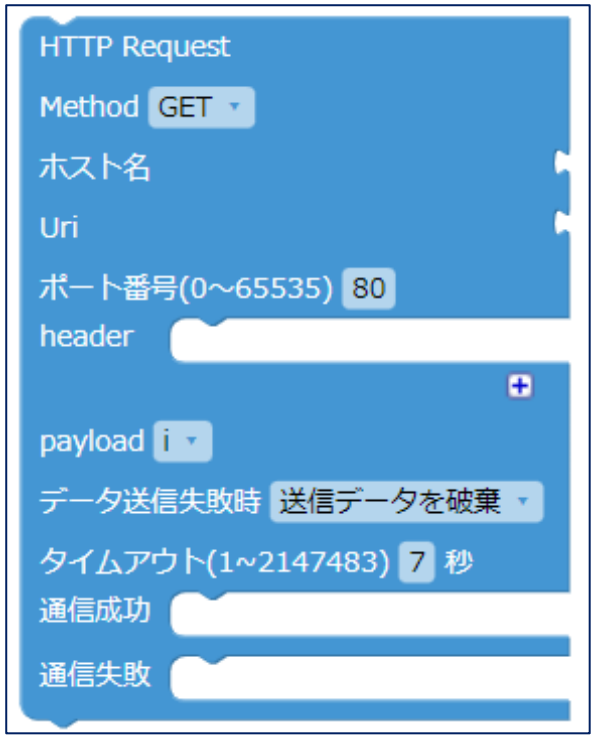


本章ではHTTPブロックの使い方を説明します。

HTTPブロックの機能説明

HTTP Requestブロックの要素を以下に説明します。

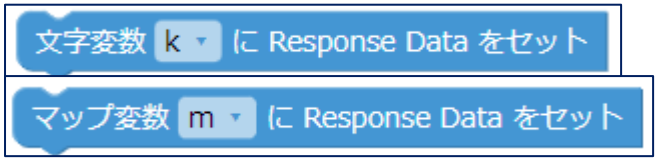
HTTPブロック



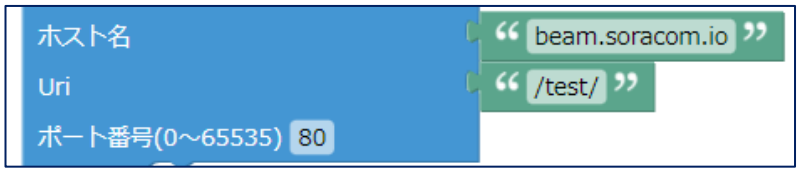
Method

MethodにはGET/POSTが選択出来ます。

GETの場合は以下のブロックでResponse Dataを取得出来ます。



ホスト名,Uri,port



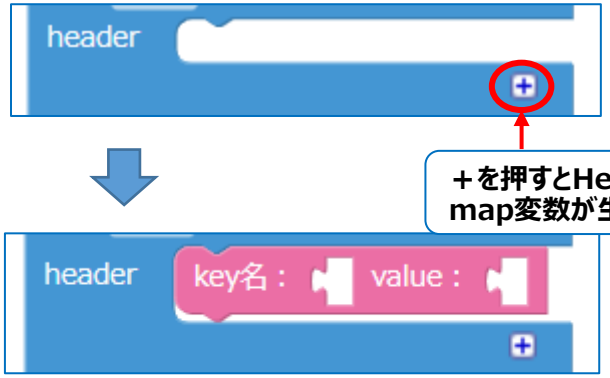
ホスト名, Uriには文字列または文字変数を接続し、
Portにはport番号を数字で入力します。

上図の例では、URL表記をすると、
http://beam.soracom.io:80/test/ になります。

HTTPブロックの機能説明

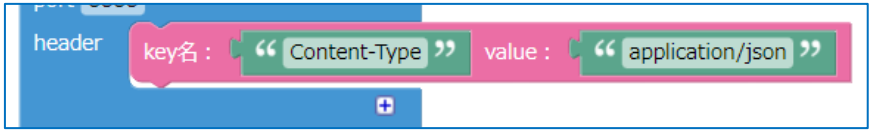
Header

httpのheaderを指定します。



＋を押すとHeaderを入力する
map変数が生成されます。

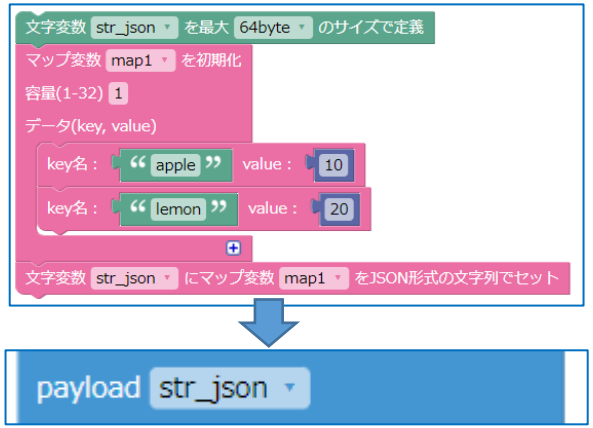
例えばContent-Typeを指定する場合は、
以下のようになります。



Payload

Payloadには文字変数を指定します。

例えば、下記の様にJson形式の文字列を準備して、payloadに設定します。

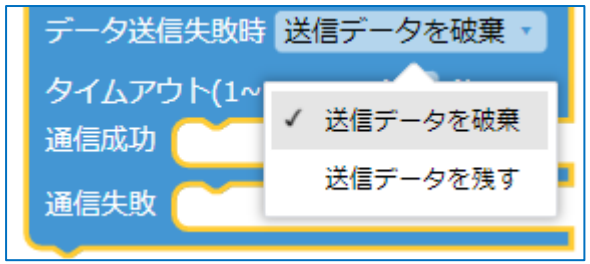


→{"apple":10,"lemon":20}

HTTPブロックの機能説明

データ送信失敗時

データ送信失敗時の振舞いを設定出来ます。



送信データを破棄

圏外時などの送信失敗時にHTTPパケットは破棄されます。

送信データを残す

送信失敗したHTTPパケットは保持され、次のHTTP requestが発行された際に送信されます。

※「送信データを残す」は現バージョンでは不備がありますので、「送信データを破棄」でご使用ください。データを残す機能は今後のアップデートで対応致します。

タイムアウト

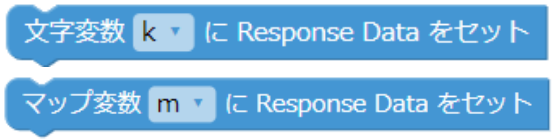
サーバー応答までのタイムアウト時間を設定出来ます。

通信成功/失敗

通信成功/失敗時の処理を設定することが出来ます。

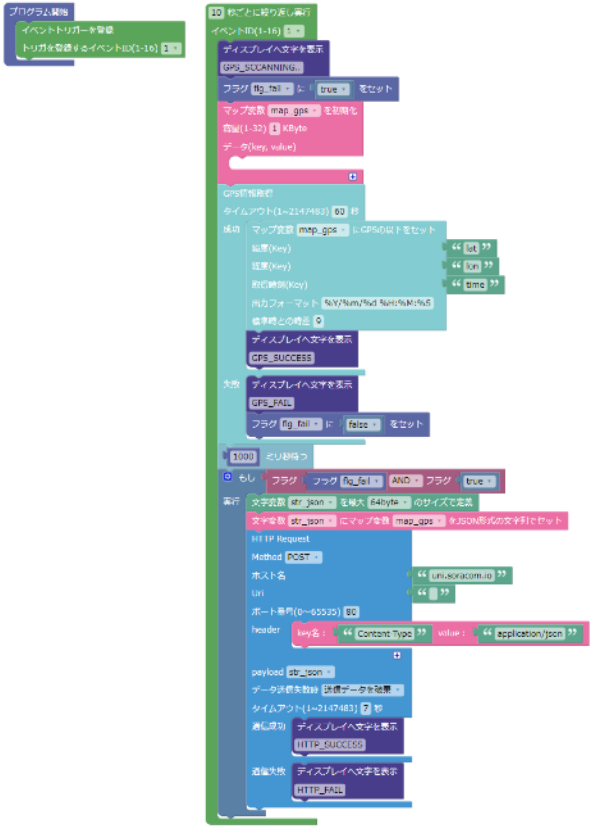
通信の結果のStatusを取得するには以下のブロック

応答のデータを取得するには以下のブロック



※現行のバージョンでは送信失敗時のReturn Status CodeやRespons Dataはセット機能していない為、今後のアップデートで対応致します。

HTTPブロックのチュートリアル



本章ではHTTPブロックの使い方を演習します。

演習レシピとしては、

- ・GPSデータ取得
- ・現在時刻の取得
- ・Jsonデータの作成
- ・HTTP送信

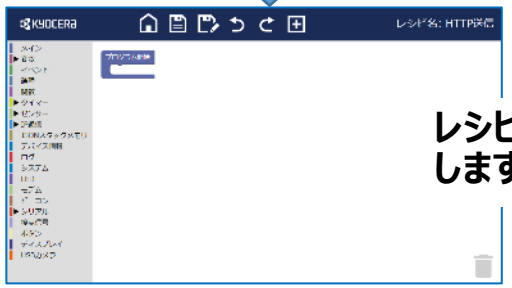
を行います。

チュートリアル① GPSの値と現在時刻を定期的にサーバに送信します。

1.レシピの新規作成

新しいレシピの作成には新規作成を押下します。

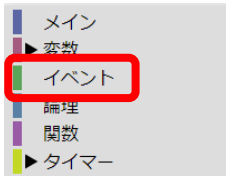
タイトルを入力して新規作成ボタンを押下します。
(説明は任意です。)

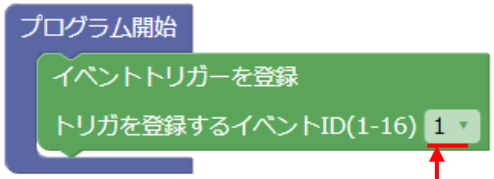


レシピ編集画面に遷移します。

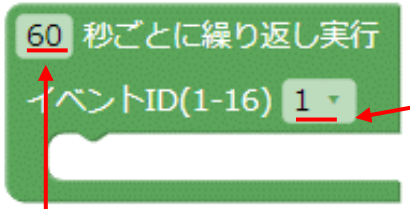
2.イベントトリガー/ブロックの配置

「プログラム開始」に「イベントトリガーブロック」をドラッグして接続します。





以下のイベントブロックを配置します。



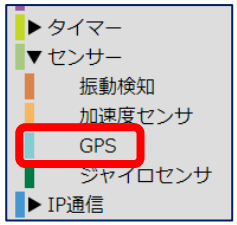
イベント番号が同じことを確認ください。

今回は60秒ごとに処理を行いますので、60秒に変更します。

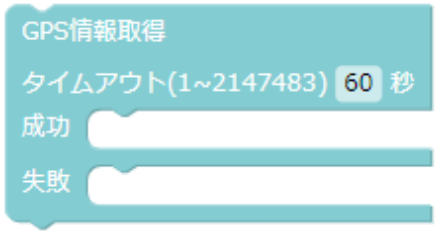
チュートリアル②

4.GPSブロックの配置

センサーカテゴリの中にあるGPSブロックを配置します。



GPSは取得するためのブロックと



取得したGPSの値と現在時刻を保存するブロックが準備されています。

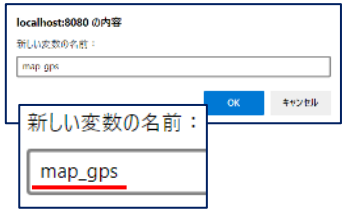
保存する先がマップ変数になります。

上記２つのブロックを配置します。

5.マップ変数を作成

取得した値を格納するマップ変数という入れ物を作ります。

map_gpsと言う変数名を作ります



マップ変数を初期化するブロックを配置し、変数名を変更します。

上記で作成した変数名を選択します。

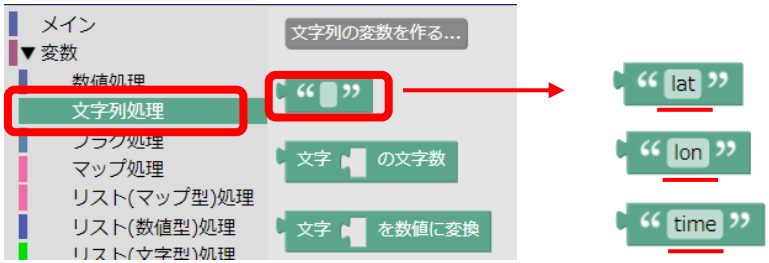
※マップ変数とは、keyとvalueで構成される２元配列変数のイメージです。

チュートリアル③

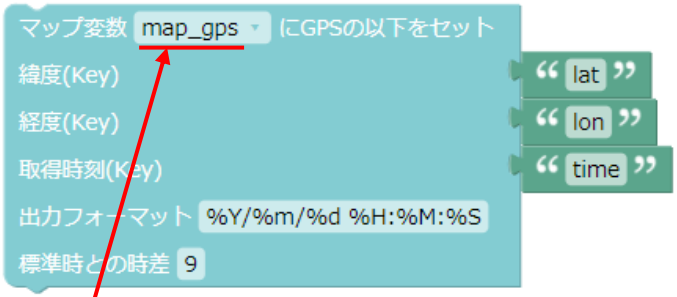
6 .JsonのKey名の作成

送信するJsonのKey名を作成します。

「文字列処理」の「固定文字列」ブロックを３つ作り、文字列として lat, lon, time を入力します。



固定文字列をGPS取得ブロックに接続していきます。

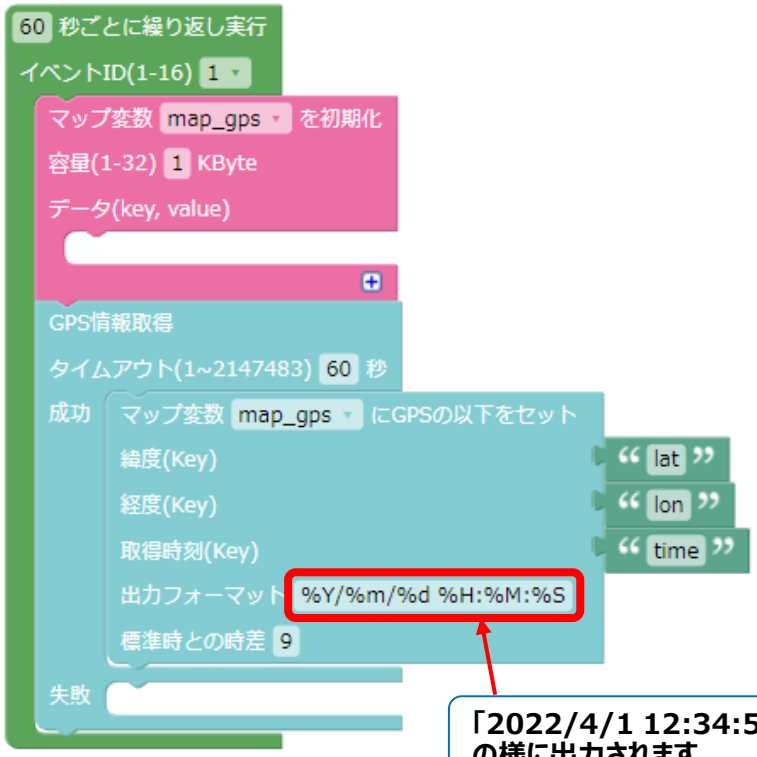


map変数名も変更します。

7 .GPSブロックの組み立て

GPSとマップ変数のブロックを組み立てます。

イベントブロックとも合わせて以下のようになります。



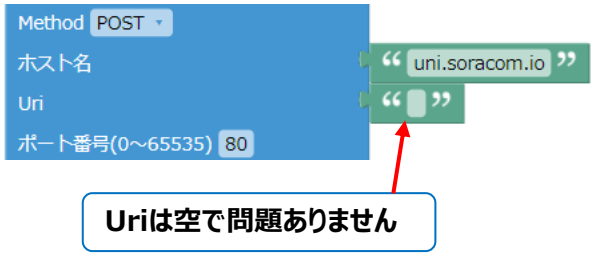
「2022/4/1 12:34:56」の様に出力されます

チュートリアル④

8.HTTPブロックの配置

本題であるHTTPブロックを配置します。

「固定文字列」でホスト名とUriを作り接続します。

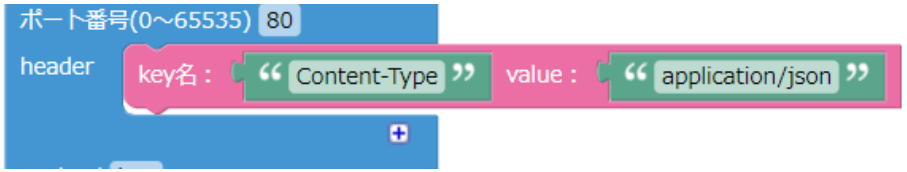


9.HTTPヘッダの作成

HTTPのヘッダを指定します。

Headerを追加するには下記「＋」を押します。

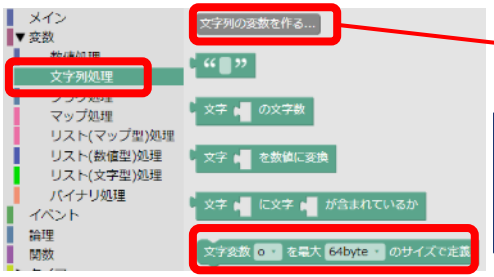
ここに「固定文字列」として、Content-Type, application/jsonを追加します。



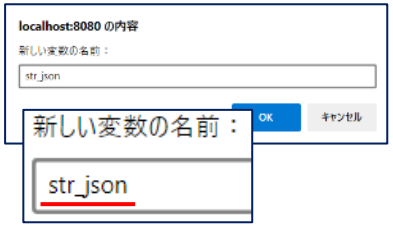
チュートリアル⑤

10.Json文字列の作成

payloadに指定するJson文字列を作成します。



str_jsonと言う
変数名を作ります



文字変数を定義するブロックを配置します

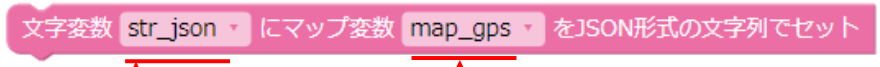
1280byteにサイズを
変更します。

作成したstr_jsonに
変更します。

※文字変数は必ず定義をしてから
使用しないとエラーとなります。

11.マップ変数からJson文字列を作成

GPSの値が入ったマップ変数をJson文字列に変換するブロックを配置します。



変換先にstr_jsonを指定

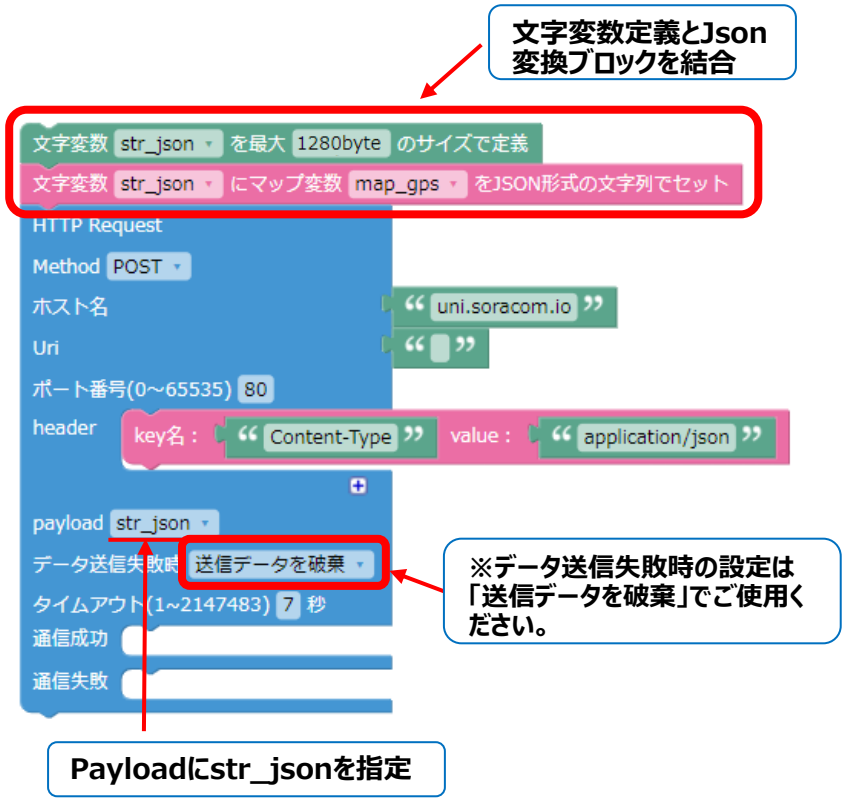
変換元にmap_gpsを指定

文字変数の定義と以下の順番で結合します。

チュートリアル⑥

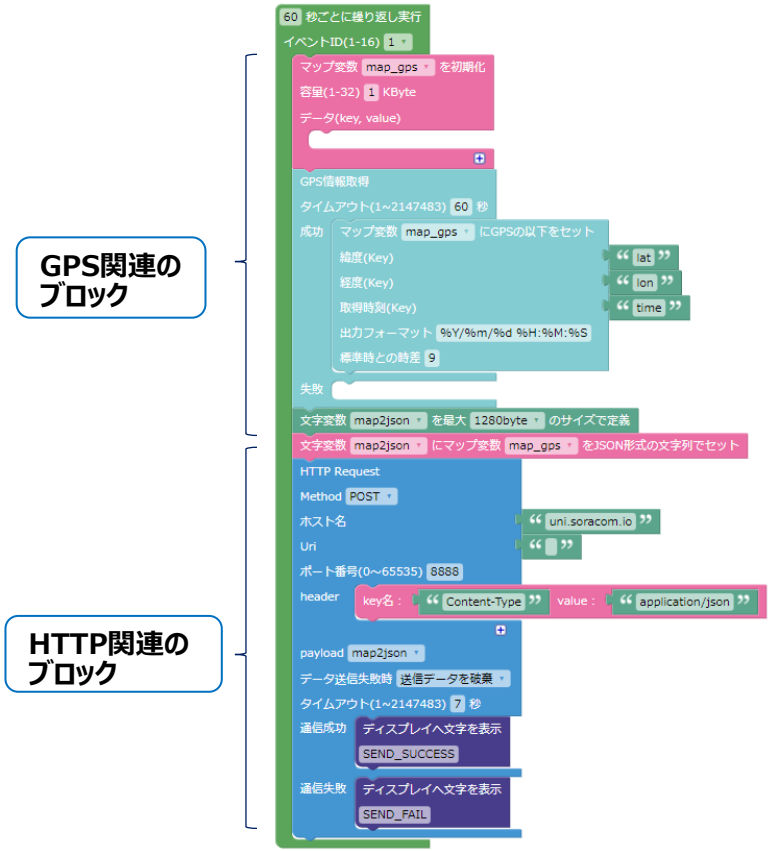
12.HTTPのpayloadの指定

作成したJson文字列をHTTPブロックへ結合し、
payloadに指定します。



13.GSPブロックとHTTPブロックの結合

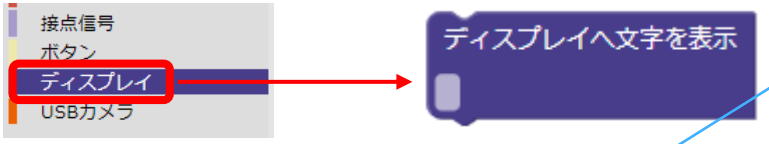
前記までのブロックを結合すると以下様になります。



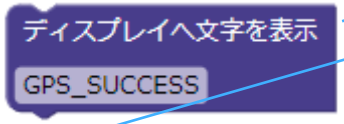
チュートリアル⑦

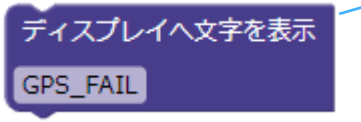
14.ディスプレイブロックの配置

前記までのブロックでも動作をしますが、端末の動作状態がわからないため、状態の表示を追加します。



ここでは以下の５つの状態表示を作成し、処理の各場所へ配置を行います。



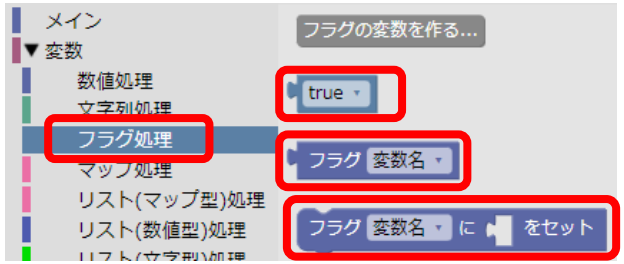


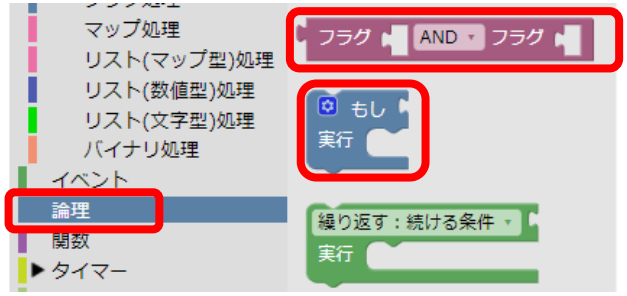
チュートリアル⑧

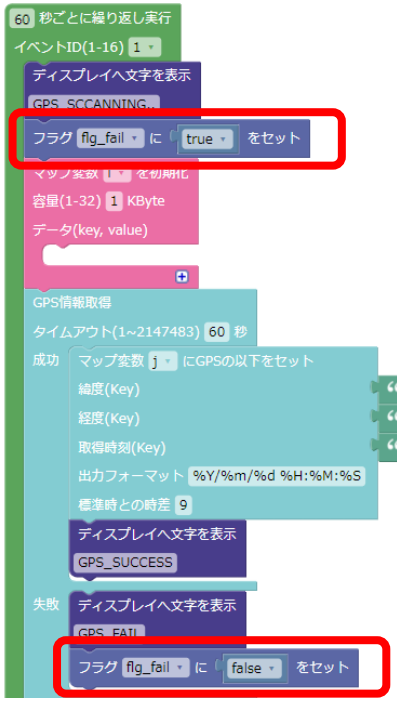
15.送信条件の追加

現状の動作では、GPS捕捉が失敗した場合においてもデータの送信を行ってしまうため、成功した場合のみ送信するよう条件を設定します。

ここでは「フラグ変数」と「論理」を使用して構成しています。下記ブロックを用いて、先のようにブロックを構成してください。







※ブロックの構成は右上の矢印に続きます。

HTTP関連ブロックは「もし」ブロックの実行の中に配置します。

※表示の変化が見やすいように１秒のウェイトを置いてます。「タイマー」「現在時刻」の中にあるブロックです。

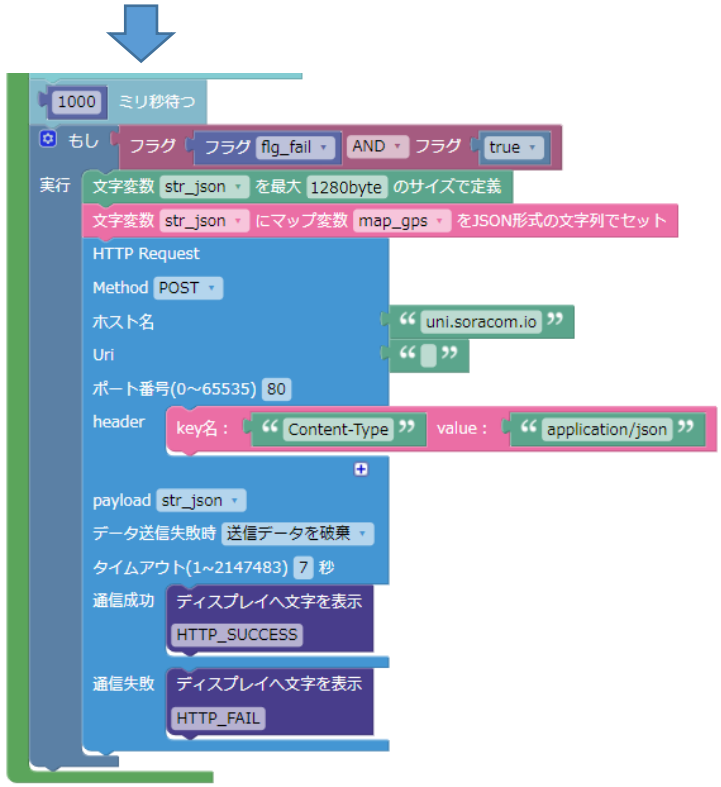
チュートリアル⑨

16.レシピの完成

前記完成したレシピは右図の様になります。次からレシピ実行ファイルの作成を行いますので、変数名等の間違いがないか、確認をしてください。

※完成したレシピの保存を忘れないようにお願いします。

※ブロックの構成は右上の矢印に続きます。



チュートリアル⑩

17.レシピの機器へのダウンロード

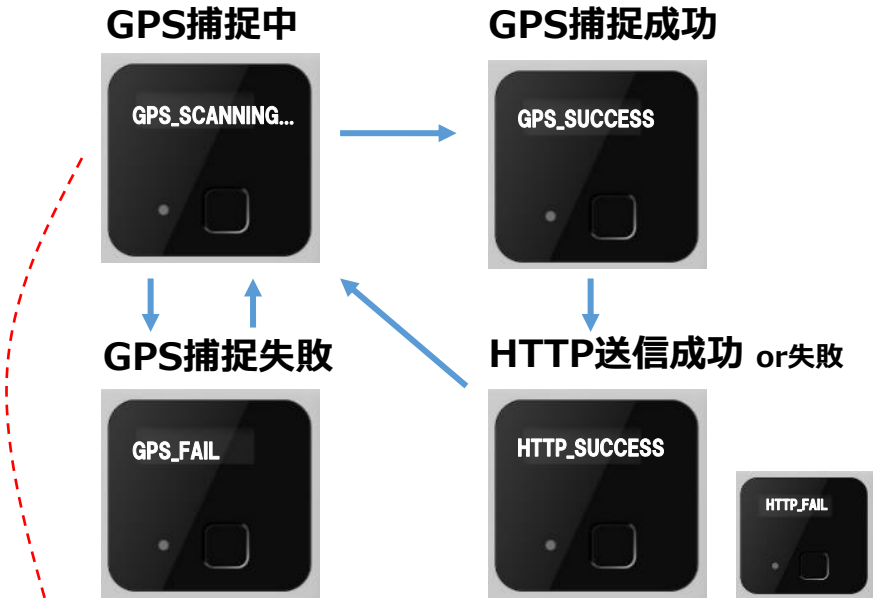
レシピ実行ファイルの作成と機器へのダウンロード方法は、「導入編」をご参照ください。

導入編

レシピ実行ファイルを機器へダウンロードする手順ページ参照

18.機器の動作

レシピをダウンロードすると、機器上でレシピが動作を開始し、GPSを捕捉に行き、以下のような画面遷移になります。



※GPSのスキャンが長い場合、表示の焼付きを防止するために、10秒程度で画面が消灯します。

チュートリアル⑪

19.送信データの確認①

送信データを確認するための設定を行います。

1-1.ログイン

SORACOMコンソールのログイン

以下のURLからログインします。

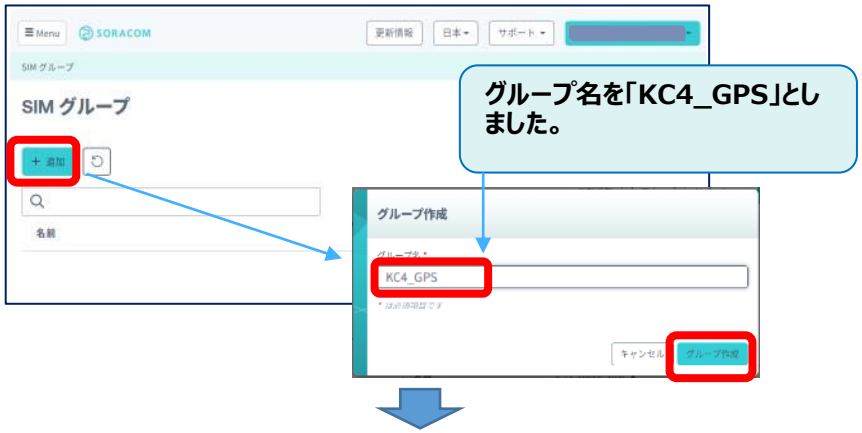
https://console.soracom.io

1-2.グループの作成①

SORACOMコンソールにログインをし、左上の「Menu」を選択し、その中の「SIMグループ」を選択します。

1-3.グループ作成②

左にある「＋追加」ボタンをクリックします。

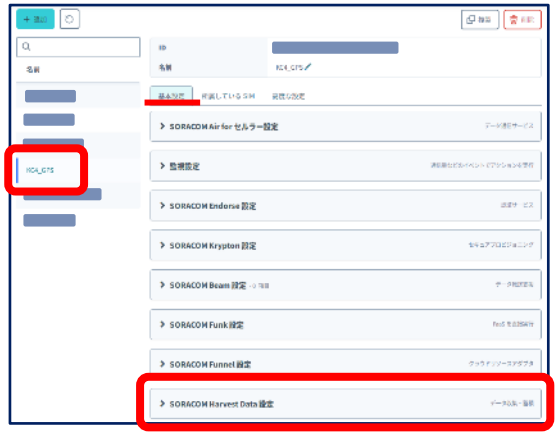


チュートリアル⑫

20.送信データの確認②

2-1. SORACOM Harvest Dataを設定

「基本設定」の中のSORACOM Harvest Dataを開いて「ON」にします。



SORACOM Harvest Data

ON

保存

2-2.設定確認

費用が発生する注意事項を確認の上、「OK」を押下します。

OK

※データのリクエスト数に伴い、利用料が発生します。

「SORACOM」ロゴをクリックしてコンソール画面に戻ります。

グループ設定が完了したらコンソール画面に戻ります。

チュートリアル⑬

2-3.SIMへのSIMグループの適用

機器に挿入した該当のSIMにSIMグループを適用します。

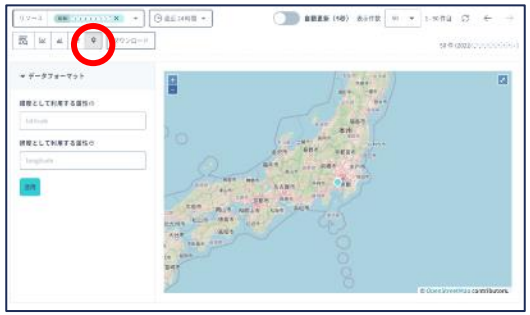
チュートリアル⑭

21. 送信データの確認

送信したデータを確認します。

対象のSIMにチェックを入れて、
「操作」-「データ確認」

送信されたGPS情報が表示されます。



マップ表示にすると地図上で位置が表示されます。

チュートリアル⑫

22. HTTPブロックチュートリアルの終了

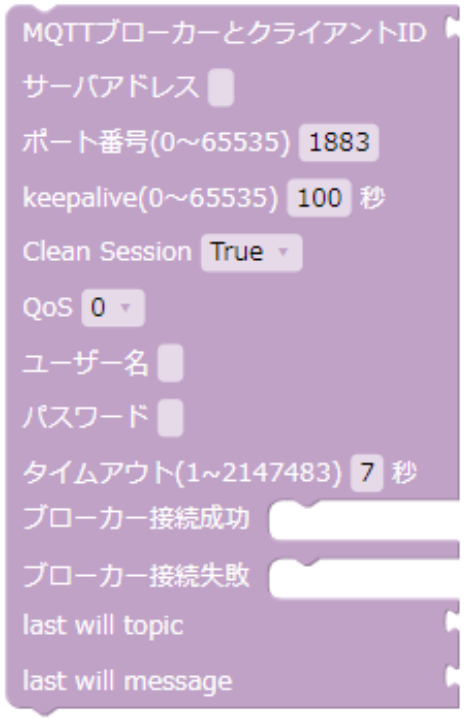
以上でHTTPのチュートリアルを終了します。

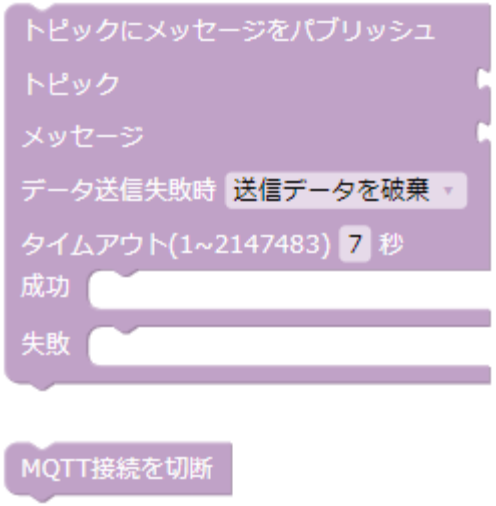
お疲れさまでした。

通信プロトコルのブロックプログラミングは少し
複雑に感じることもありますが、慣れてしまえば
今回のレシピも数分で組むことが出来ます。

他にもいろんな機能ブロックがありますので、
オリジナルレシピの作成にトライしてみてください。

MQTTブロックの使い方



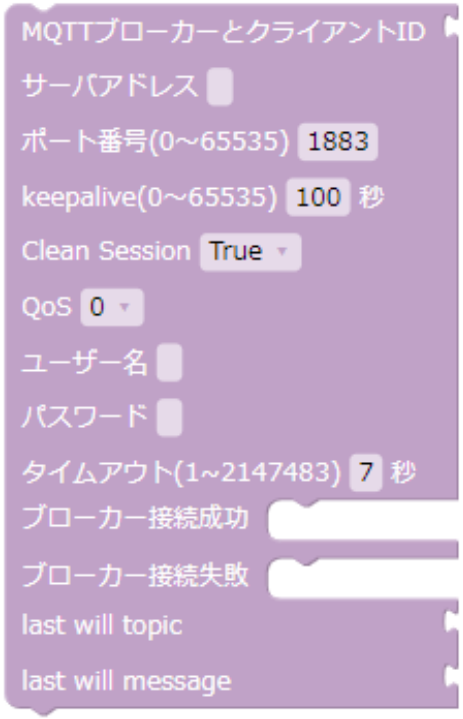


本章ではMQTTブロックの使い方を説明します。

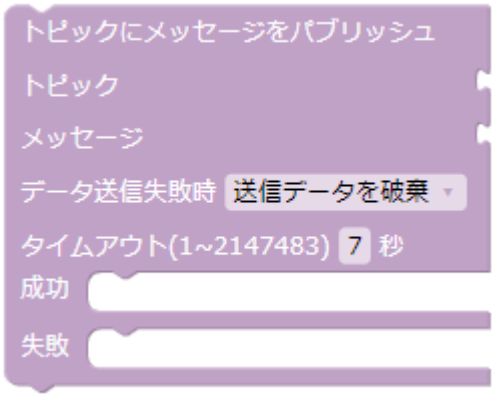
MQTTブロックの機能説明

MQTTとパブリッシュブロックの要素を以下に説明します。

MQTTブロック



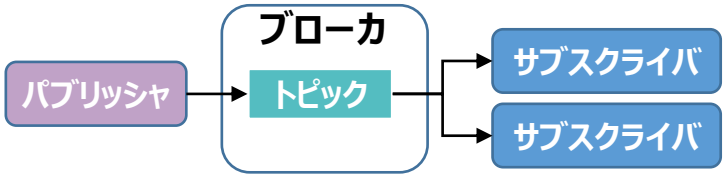
パブリッシュブロック

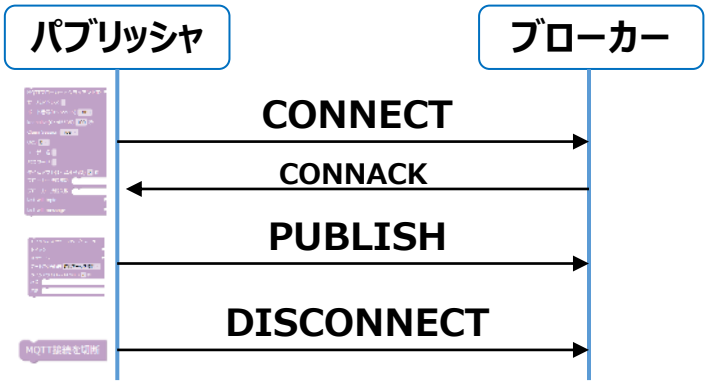


切断ブロック

MQTTプロトコル

本機器のMQTTはV3.1.1相当となり、
機能はパブリッシャのみ対応しています。





MQTTブロックの機能説明

MQTTブロック説明

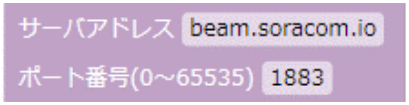
クライアントID

クライアントIDを文字列で指定します。

※クライアントIDを空文字  にすることも可能ですがこの場合はサーバー側で任意文字が割り振られる場合の他、エラーとなる場合があります。
(また、クライアントから同じSessionへの接続は出来なくなります。)

サーバアドレス/ポート番号

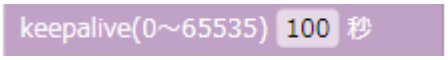
接続するサーバのアドレスとポート番号を指定します。



※テスト用に公開されている「test.mosquito.org」などのブローカーでも接続確認は出来ます。
MQTTのポート番号は一般的に1883が使われますがサーバーの指定に従ってください。

Keepalive

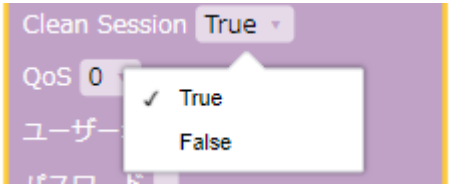
Sessionの継続時間を指定します。



※この時間の間に同クライアントIDでの接続か、切断処理をしない場合、サブスクライバにWill(遺言)が発行されます。

Clean Session

既存Sessionを破棄するかどうかを指定します。



True:既存セッションを破棄し新しいセッションで接続します。
False:既存のセッションで接続します。
※Trueで接続した場合、事前にQoS=1で送信したメッセージをサブスクライバが受け取って無い時に、ブローカー側でそのメッセージがクリアされます。

MQTTブロックの機能説明

MQTTブロック説明

QoS

Quarity of Serviceを指定します。

QoS0：メッセージの到着確認は行わない。
QoS1：メッセージの到着確認を行う。(パブリッシャ-ブローカー間)
QoS2：サブスクライバまでの到着確認を行う。
(チュートリアルでのAWSはQoS1までの対応となります。)

ユーザー名/パスワード

ユーザー名/パスワード認証を行うサーバに対しての設定を行います。

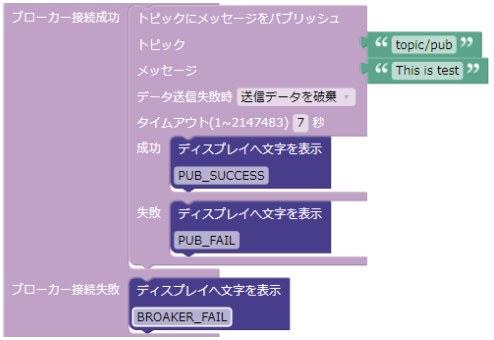
タイムアウト

ブローカー応答までのタイムアウト時間を設定します。

通信成功/失敗

ブローカーへの通信成功/失敗時の処理を設定することが出来ます。

通常は以下の様にこの中にパブリッシュブロックを配置します。

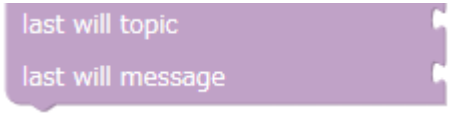


MQTTブロックの機能説明

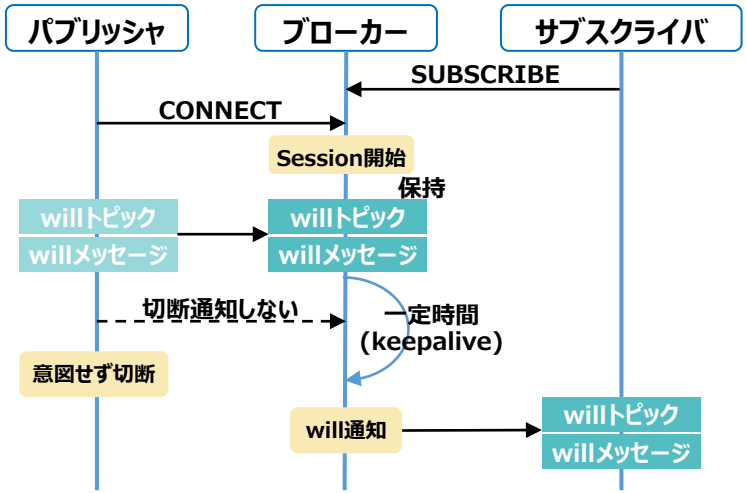
MQTTブロック説明

Last will

willトピックとメッセージを指定します。



willはパブリッシャが意図せずに切断された場合等に、サブスクライバに通知を出す機能となります。パブリッシャ側が正しく切断した場合は通知は出力されません。

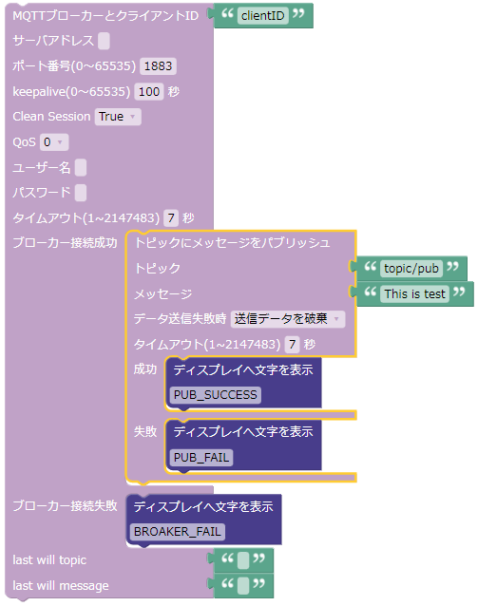


パブリッシュブロック説明

パブリッシュブロック

パブリッシュブロックはブローカー接続成功の中に配置して、対象トピックにメッセージを送信します。

例としては下記の様に「ブローカー接続成功」の中にパブリッシュブロックを配置してメッセージを送信します。



MQTTブロックの機能説明

パブリッシュブロック説明

トピックとメッセージ

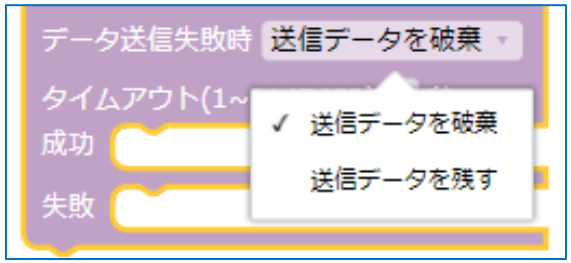
パブリッシュするトピックとメッセージを指定します。

上記の例では、「topic/pub」の階層構造の中に「This is test」メッセージを送信する形になります。

実際には、Json形式のメッセージなどを送信する形になるので、ブロックとして前段でJson形式の文字列を作成して接続します。

データ送信失敗時

データ送信失敗時の振舞いを設定出来ます。



送信データを破棄

送信失敗時にトピックとメッセージのパケットは破棄されます。

送信データを残す

送信失敗したトピックとメッセージのパケットは保持され、次のパブリッシュリクエストが発行された際に送信されます。

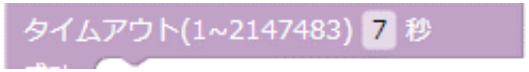
※「送信データを残す」は現バージョンでは不備がありますので、「送信データを破棄」でご使用ください。データを残す機能は今後のアップデートで対応致します。

MQTTブロックの機能説明

パブリッシュブロック説明

タイムアウト

サーバー応答までのタイムアウト時間を設定出来ます。



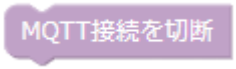
通信成功/失敗

パブリッシュ成功/失敗時の処理の設定が出来ます。

切断ブロック説明

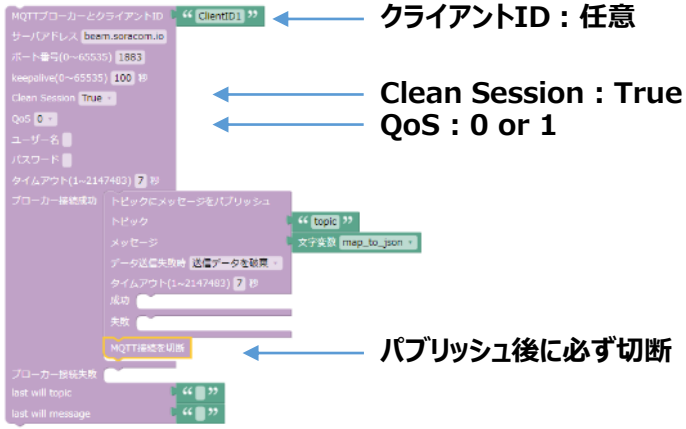
MQTT接続を切断

ブローカーに対して切断命令を発行します。

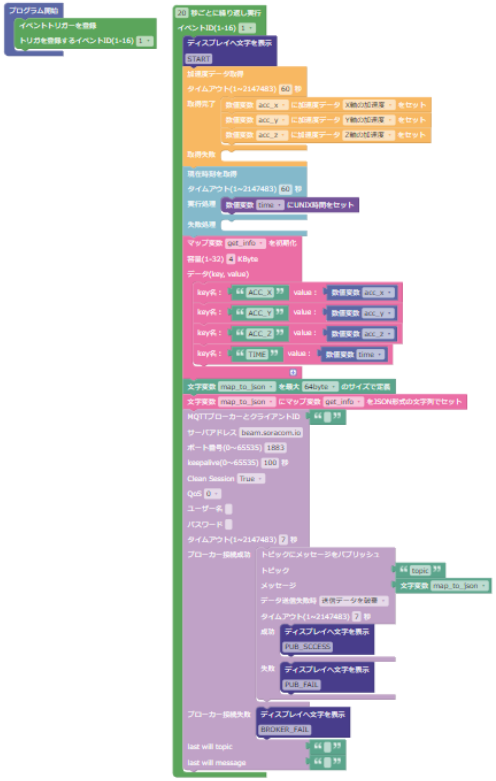


Last willの説明のところにも記載がありますが、接続したセッションは明示的に切断しないと、サブスクライバにwill通知が発行されます。

単にデータを送信してサーバーへ集めるなどのケースでは、以下の設定で必ずパブリッシュ後に切断する処理が良いと考えます。



MQTTブロックのチュートリアル



本章ではMQTTブロックの使い方を演習します。

演習レシピとしては、

・加速度の傾きデータ、現在時刻を取得

・Json形式でMQTT送信

を行います。

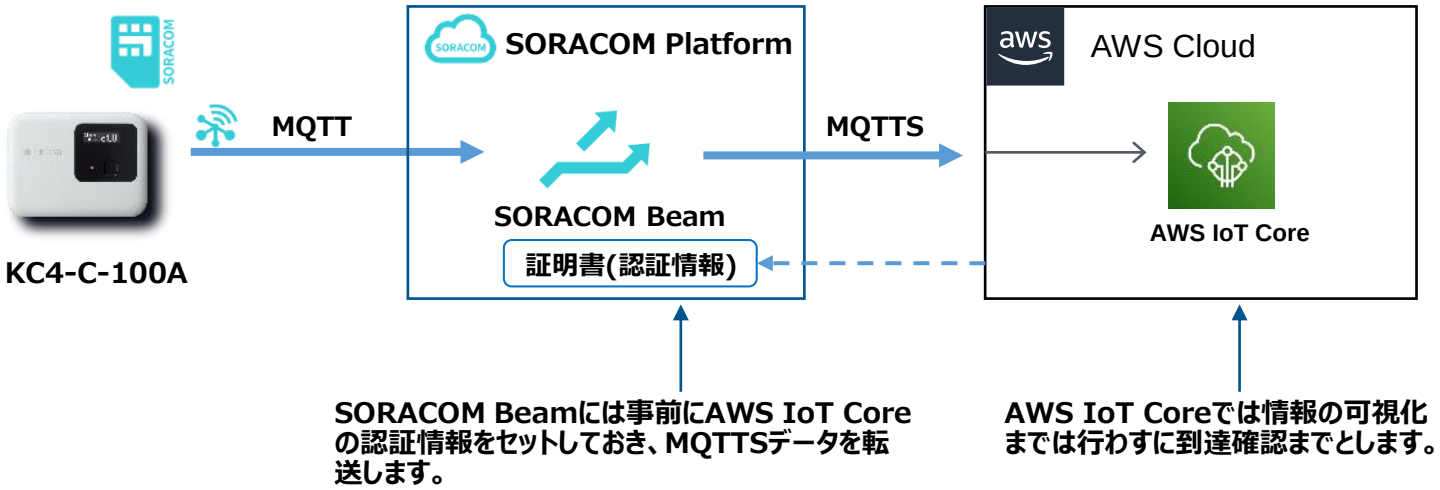
※本チュートリアルではSORACOM IoT SIMに加えて、AWSも利用しますので、AWSのアカウントが必要となります。

チュートリアル①

1.データの流れ

本チュートリアルでは、２つのクラウドサービスを利用します。

1. SORACOM（IoT SIMとSORACOM Beamの利用）
2. AWS (MQTTS受信で利用)



チュートリアル②

1.AWSでの受信準備

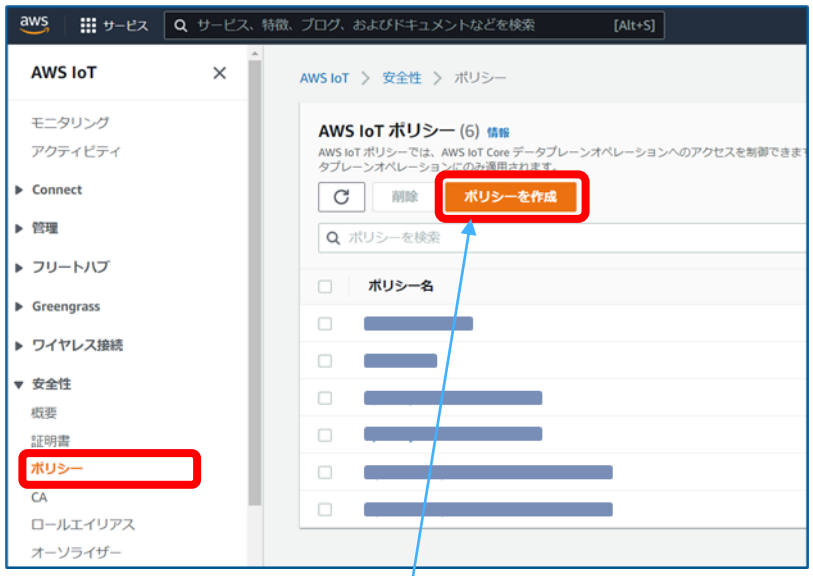
AWS IoT CoreでMQTTSのパケットを受信できるように準備を行います。

AWSコンソールにログインし、左上の「サービス」を選択して、その中の「IoT」を選択し、右に出るメニューの「IoT Core」を選択します。

1-1.ポリシーの作成①

まず、ポリシーを作成します。

AWS IoTメニューの「安全性」の「ポリシー」を選択すると、右側にポリシー一覧が現れます。
（※初めての場合はポリシーのリスト空となります。）

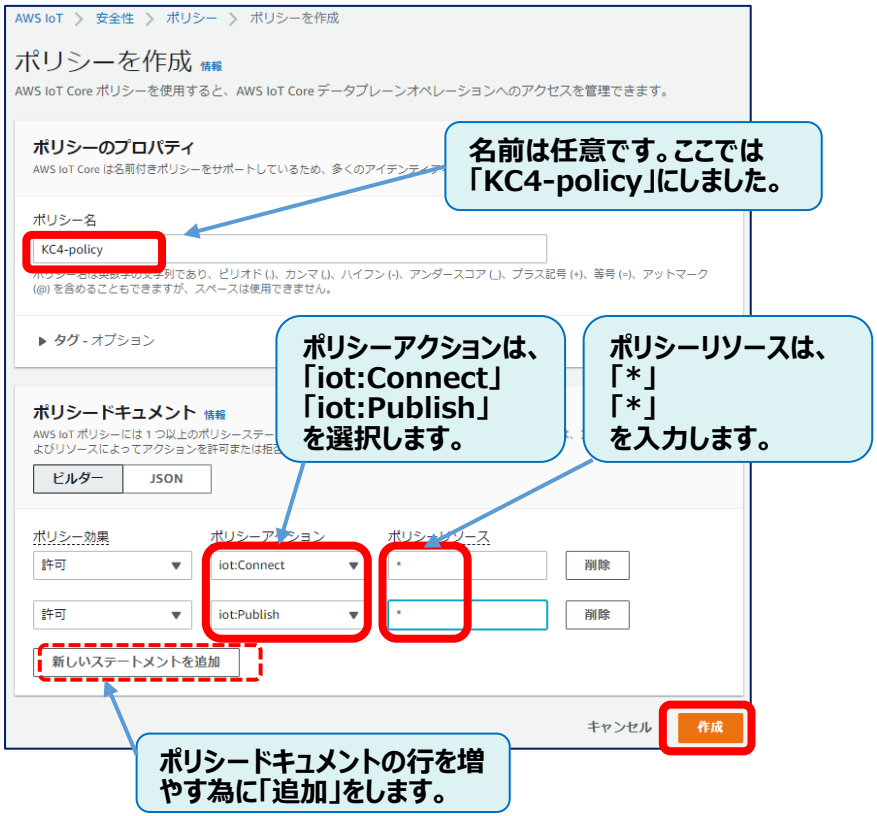


「ポリシーを作成」ボタンをクリックします。

チュートリアル③

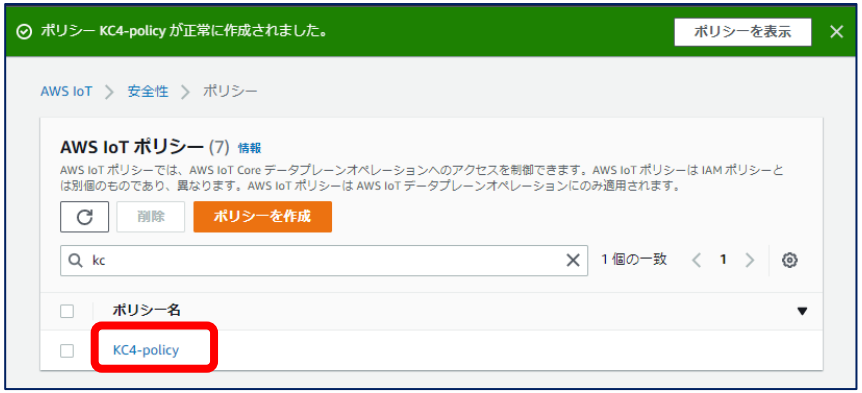
1-2.ポリシーの作成②

ポリシー名、ポリシーアクション、ポリシーリソースの入力/選択を行い、「作成」をクリックします。



1-3.ポリシーの作成③

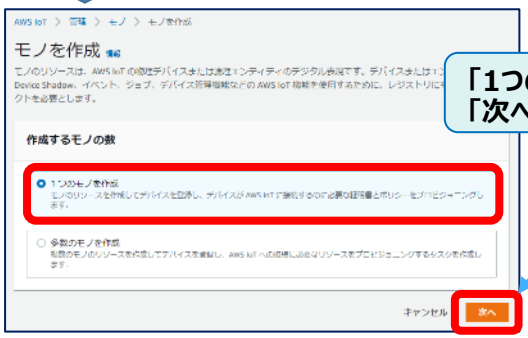
ポリシー作成が完了すると、以下のように正常に作成されましたの表示と、該当のポリシーが表示されます。



チュートリアル④

1-4.モノの作成①

AWS IoTメニューの「管理」の「モノ」を選択すると、右側にモノの一覧が現れます。
(※初めての場合はモノのリストは空となります。)



1-5.モノの作成②

「モノの名前」を入力してモノの作成をします。
追加設定等は特に変更しません。

チュートリアル⑤

1-6.デバイス証明書の作成

モノの作成の流れで「デバイス証明書」の作成を行います。

「新しい証明書」を選択し「次へ」ボタンをクリックします。

1-7.証明書にポリシーをアタッチ

作成した「デバイス証明書」にポリシーの関連付けを行います。

先程作成したポリシー「KC4-policy」を選択し「モノを作成」ボタンをクリックします。

チュートリアル⑥

1-8.証明書とキーのダウンロード

作成された証明書とキーのファイルをダウンロードします。

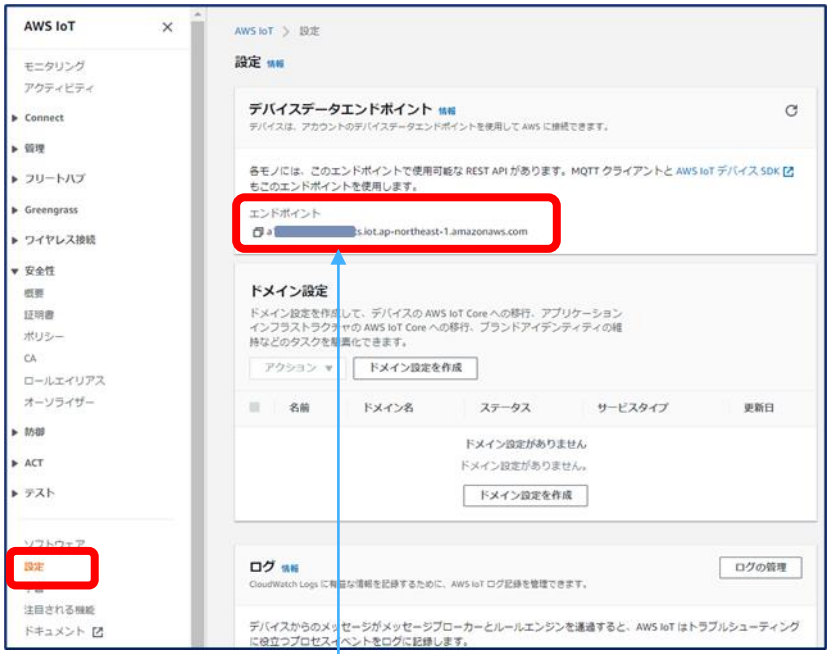
この４つをダウンロードします。

※CA3はチュートリアルでは使用しないので、ダウンロードは任意です。

ダウンロードが完了したら「完了」をクリックします。

1-9.エンドポイントの確認

機器からAWSへアクセスするためのエンドポイントを確認します。左のメニュの「設定」を選択します。

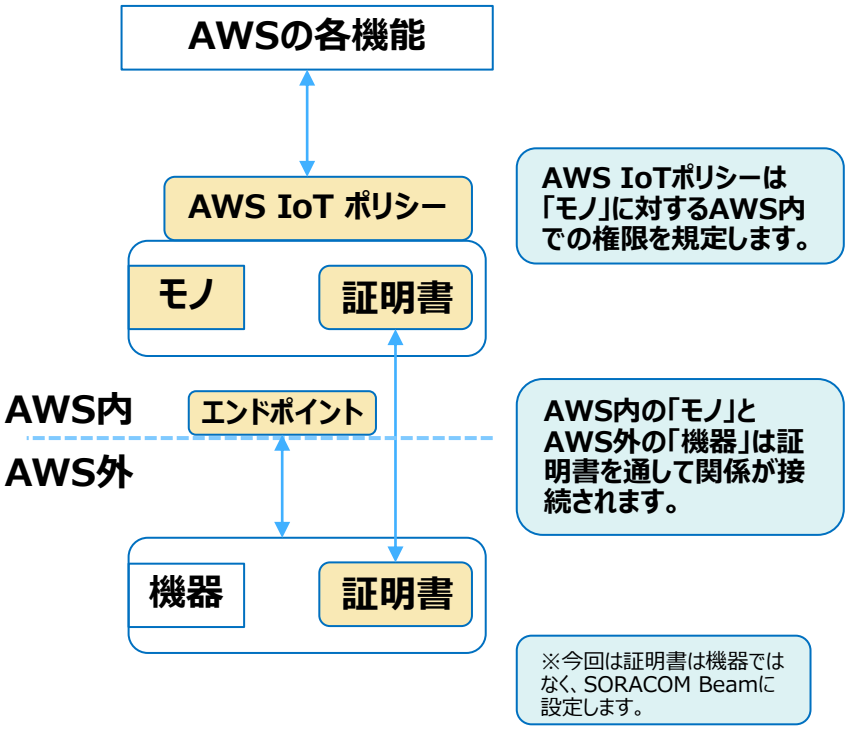


エンドポイントが表示されたら、このエンドポイントをコピーしておきます。

チュートリアル⑥

1-10.モノ,証明書,ポリシーの関係

作成した「モノ」、「証明書」、「ポリシー」の関係は以下の様になります。



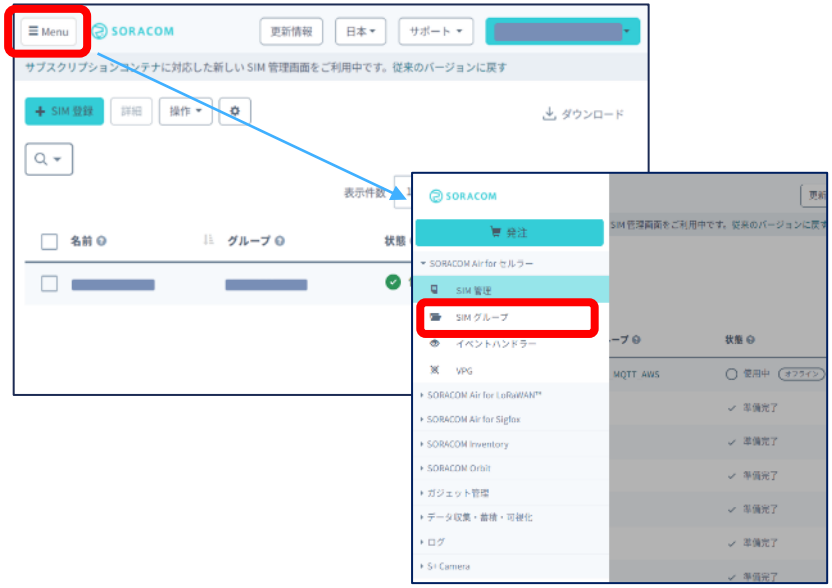
チュートリアル⑦

2.SORACOMでの転送準備

MQTTをMQTTSに変換してAWSに送るための
SORACOMの設定を行います。

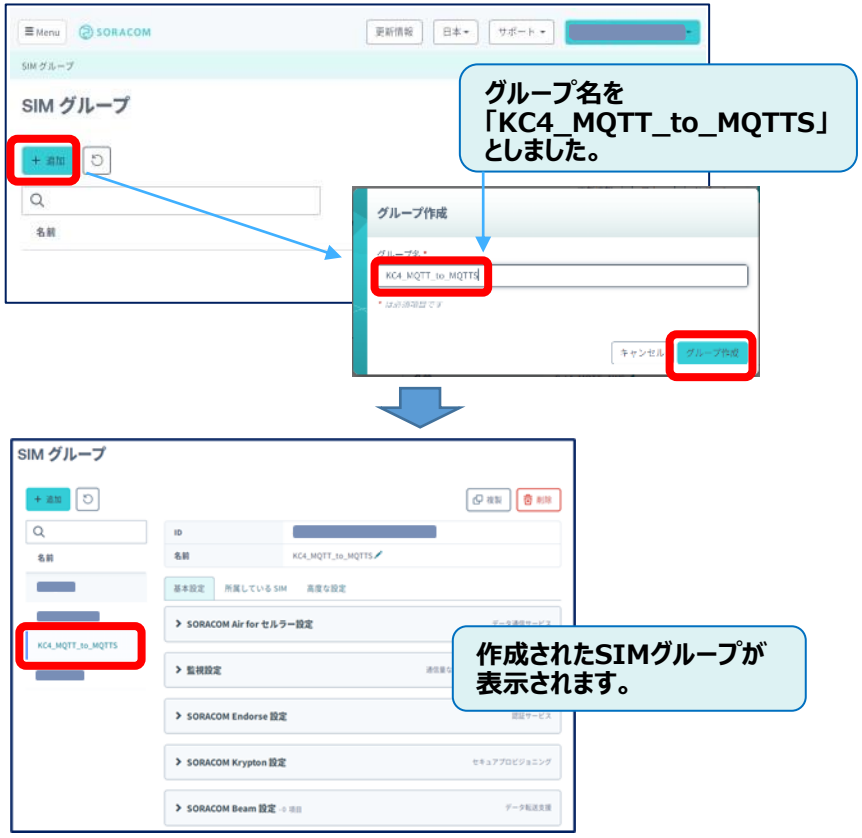
2-1.グループの作成①

SORACOMコンソールにログインをし、左上の「Menu」
を選択し、その中の「SIMグループ」を選択します。



2-2.グループ作成②

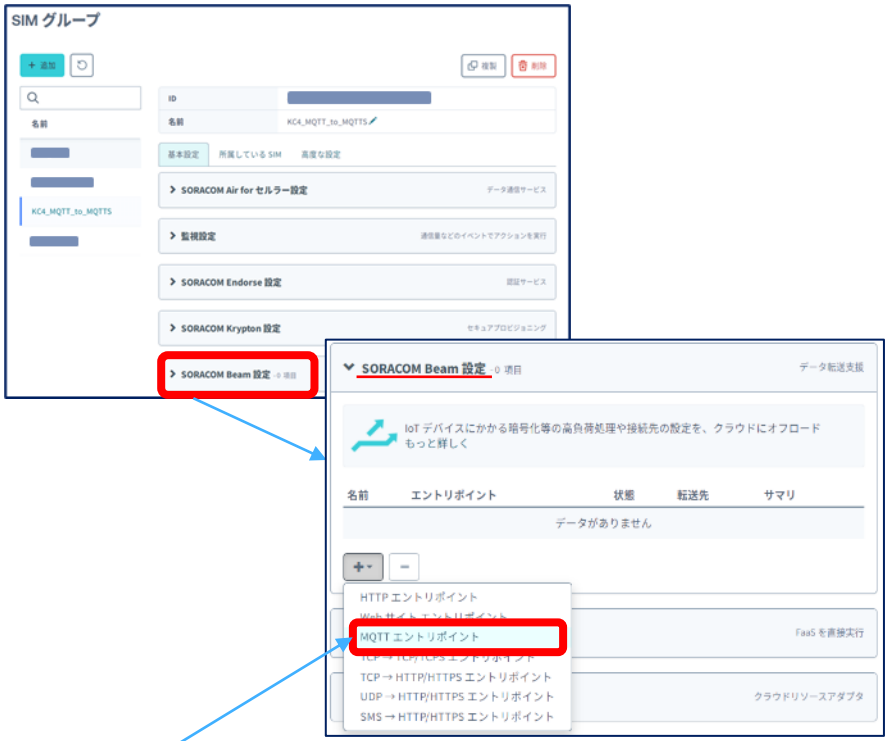
左にある「＋追加」ボタンをクリックします。



チュートリアル⑧

2-3.SORACOM Beam設定①

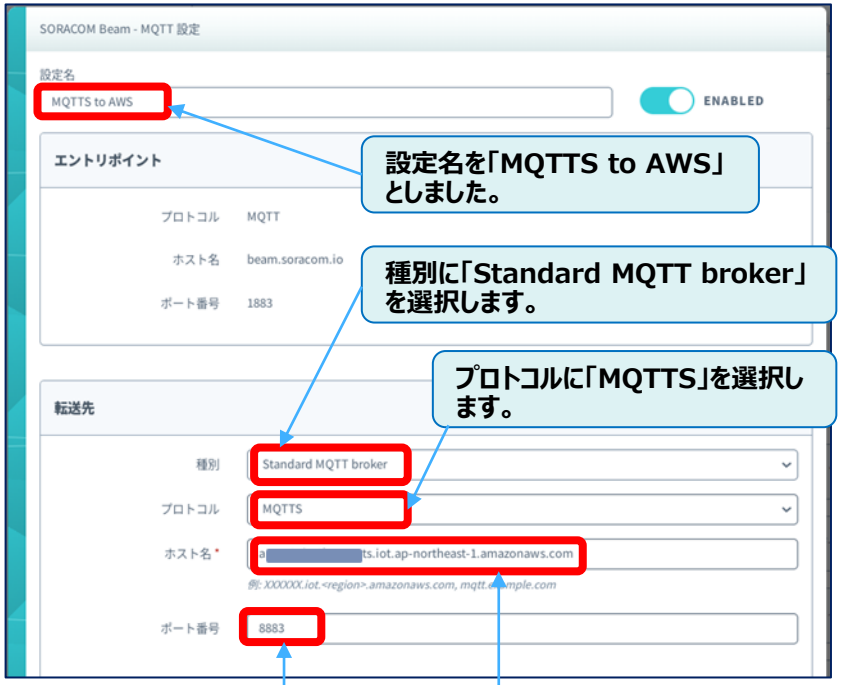
「基本設定」の中の「SORACOM Beam設定」をクリックして、展開します。



「＋」をクリックし、選択肢の中から「MQTTエントリポイント」を選びます。

2-4.SORACOM Beam設定②

「SORACOM Beam - MQTT設定」を以下のように設定します。



設定名を「MQTTS to AWS」としました。

種別に「Standard MQTT broker」を選択します。

プロトコルに「MQTTS」を選択します。

ポート番号は「8883」を指定します。

1-9項.でコピーしたAWSのエンドポイントをペーストします。

チュートリアル⑨

2-5.SORACOM Beam設定③

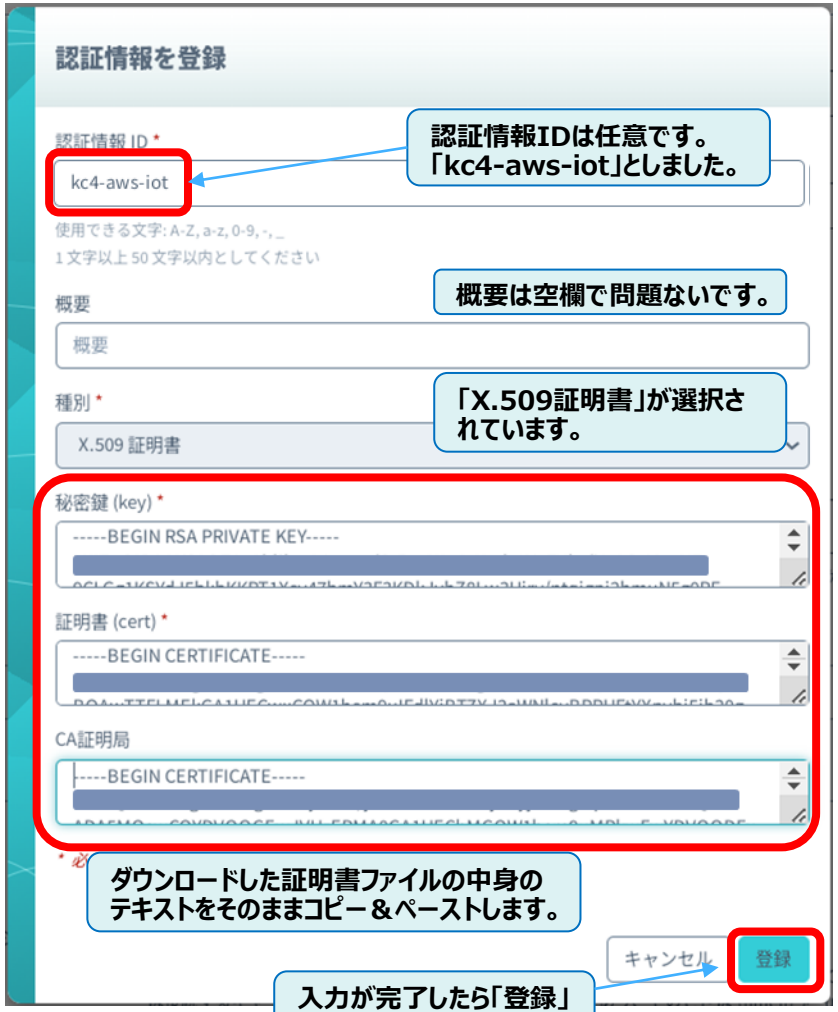
「SORACOM Beam - MQTT設定」の中段の「証明書」をONにして、その右にある「＋」をクリックします。

ダウンロードした証明書ファイルを準備します。
※ファイル名の文字列に注目します。

CA証明局

証明書(cert)

秘密鍵(key)



認証情報IDは任意です。「kc4-aws-iot」としました。

概要は空欄で問題ないです。

「X.509証明書」が選択されています。

ダウンロードした証明書ファイルの中身のテキストをそのままコピー＆ペーストします。

入力が完了したら「登録」をクリックします。

チュートリアル⑩

2-6.SORACOM Beam設定④

「SORACOM Beam - MQTT設定」の下段のIMSI付与を「ON」にして「保存」をクリックします。

2-7.SORACOM Beam設定⑤

「SORACOM Beam 設定」に先程の設定が反映されている事を確認します。

「SORACOM」ロゴをクリックしてコンソール画面に戻ります。

チュートリアル⑪

2-8.SIMへのSIMグループの適用

機器に挿入した該当のSIMにSIMグループを適用します。

チュートリアル⑫

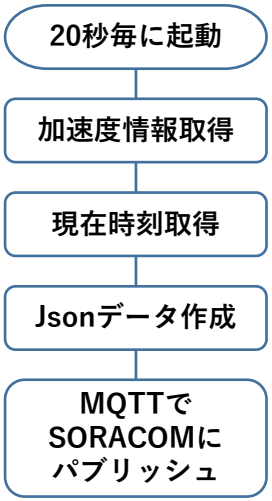
3.レシピブロックの構成

ここから機器側のレシピを作成していきます。

レシピの新規作成やブロックの配置は、導入編やHTTPのチュートリアルをご参照ください。

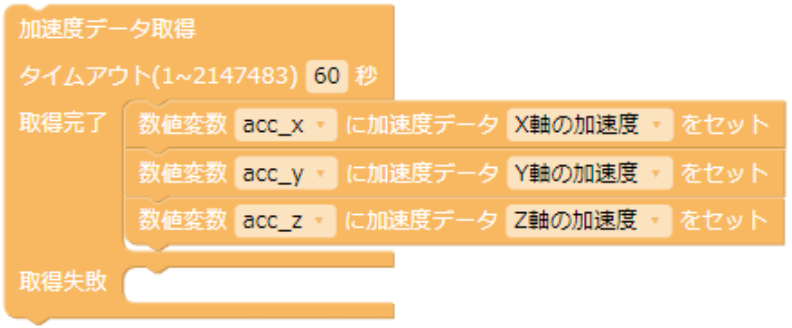
3-1.レシピの構成

本レシピは以下の流れで構成します。



3-2.加速度/現在時刻情報の取得

加速度ブロックを使用して、XYZの数値を取得

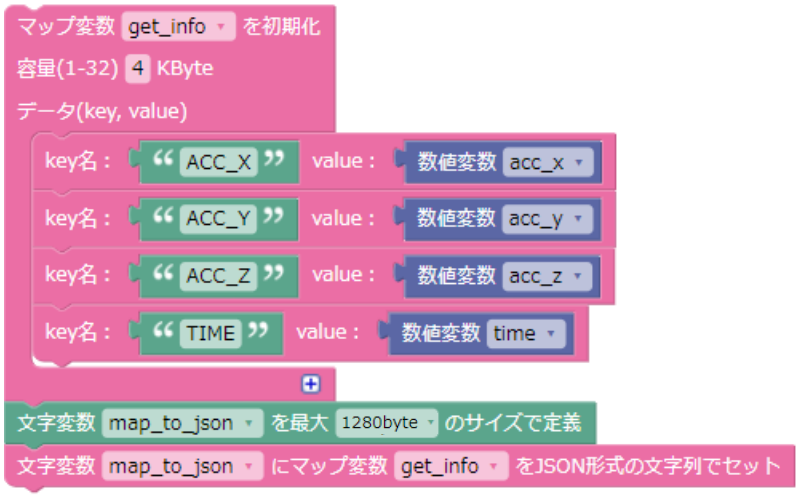


現在時刻ブロックを使用して時刻を取得

チュートリアル⑬

3-3. 取得情報のJson文字列化

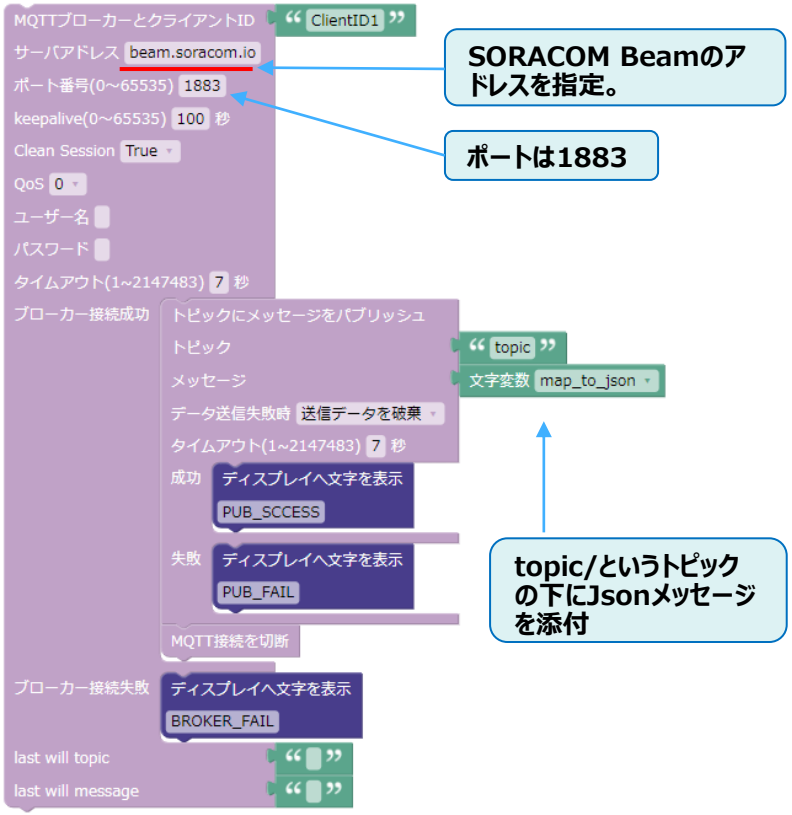
マップ変数で構成したデータをJson文字列に変換します。



マップ変数にて、keyとvalueのデータ構成を行い、その構成を文字列に置き換える処理を行います。

3-4. MQTT送信

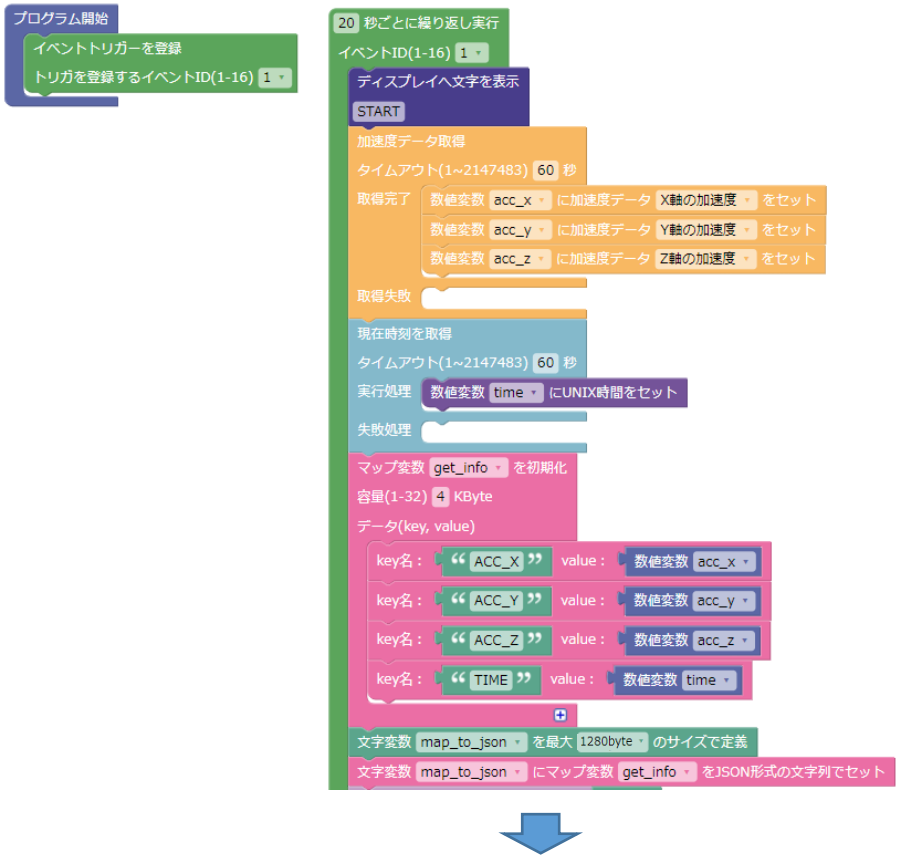
MQTTパケットをSORACOM Beam宛に送信を行います。

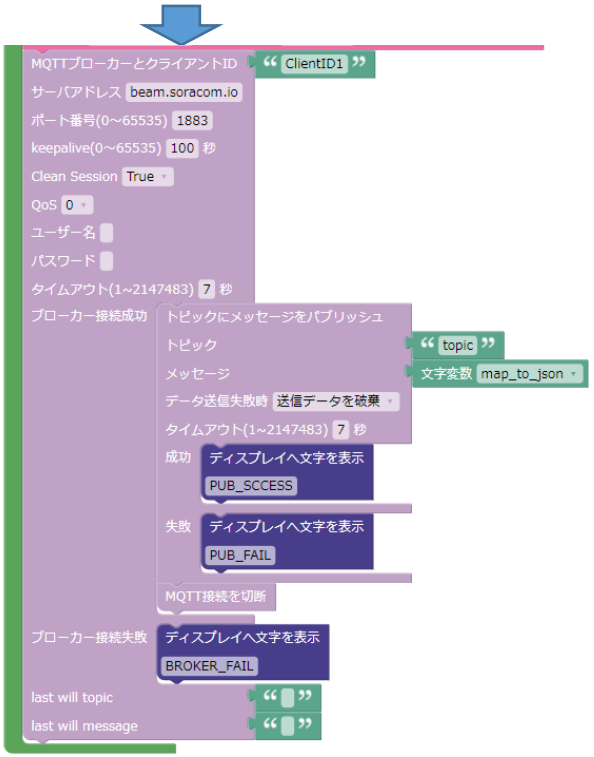


チュートリアル⑭

3-5. レシピの全体

構成したレシピ全体は以下の通りになります。





レシピを構成したら、機器への書込みを行い動作確認を行います。

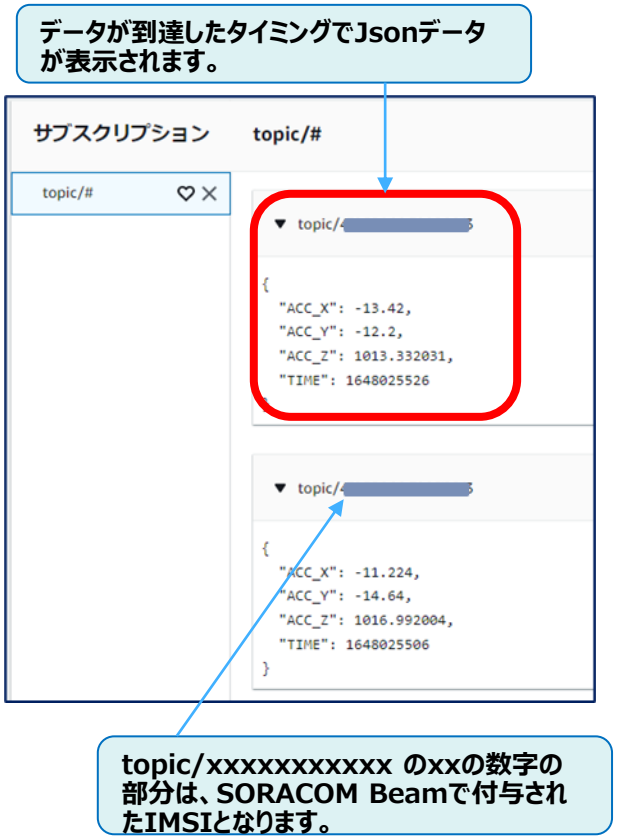
機器への書込みは「導入編」等ご参照お願い致します。

チュートリアル⑮

4.MQTTのAWS到達確認

SORACOM Beam経由でAWSにMQTTパケットが到達している事を確認します。

AWSのコンソールにて以下を選択。



チュートリアル⑯

5. MQTTブロックチュートリアルの終了

以上でMQTTのチュートリアルを終了します。

お疲れさまでした。

今回は２つのクラウドサービスを利用したため、少し複雑な設定が必要でしたが、MQTTブロックへの設定項目は少なく、シンプルに使えたと思います。

一度、クラウド側にデータが届くことが確認出来たあとは、他のセンサーなどのデータを追加することも簡単ですので、他の機能と共に色んなレシピを試してみてください。

THE NEW VALUE FRONTIER

KYOCERA

京セラ株式会社

※ 本資料は2022年6月現在のものです。
※ LTEは、ETSIの商標です。
※ ソラコム、SORACOM、又はソラコムの商品・サービス名称等は、
株式会社ソラコムまたはその関連会社の商標または登録商標です。
※ Amazon Web Services、アマゾン ウェブ サービス、AWS、
AWS IoT Coreは、米国その他の諸国における、Amazon.com, Inc.
またはその関連会社の商標です。