家族の「ただいま」を
教えてくれるキーホルダー

高橋庸介

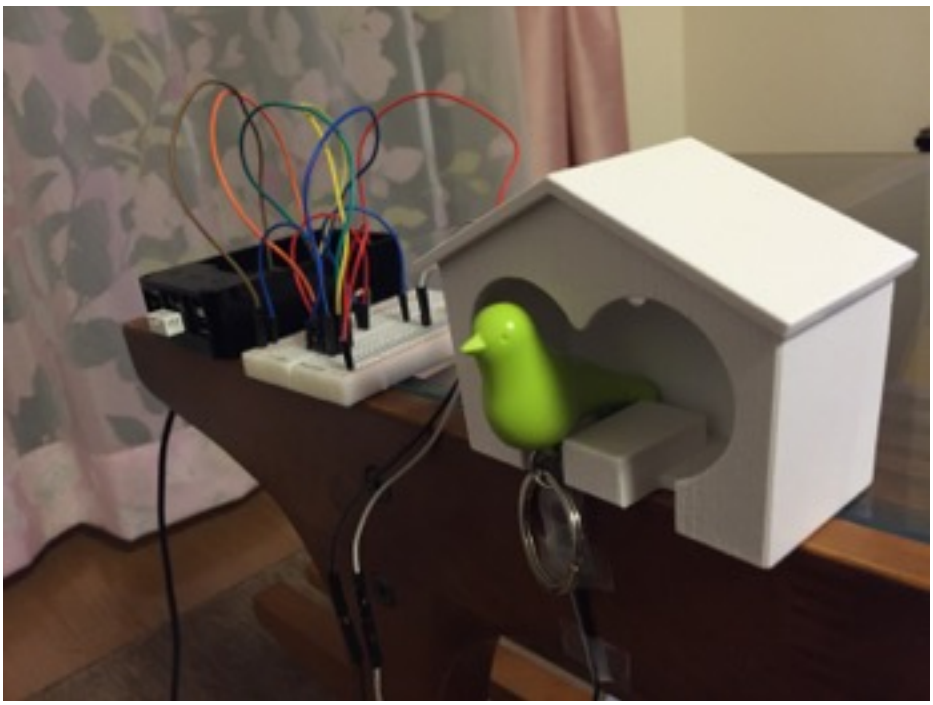
目次

概要

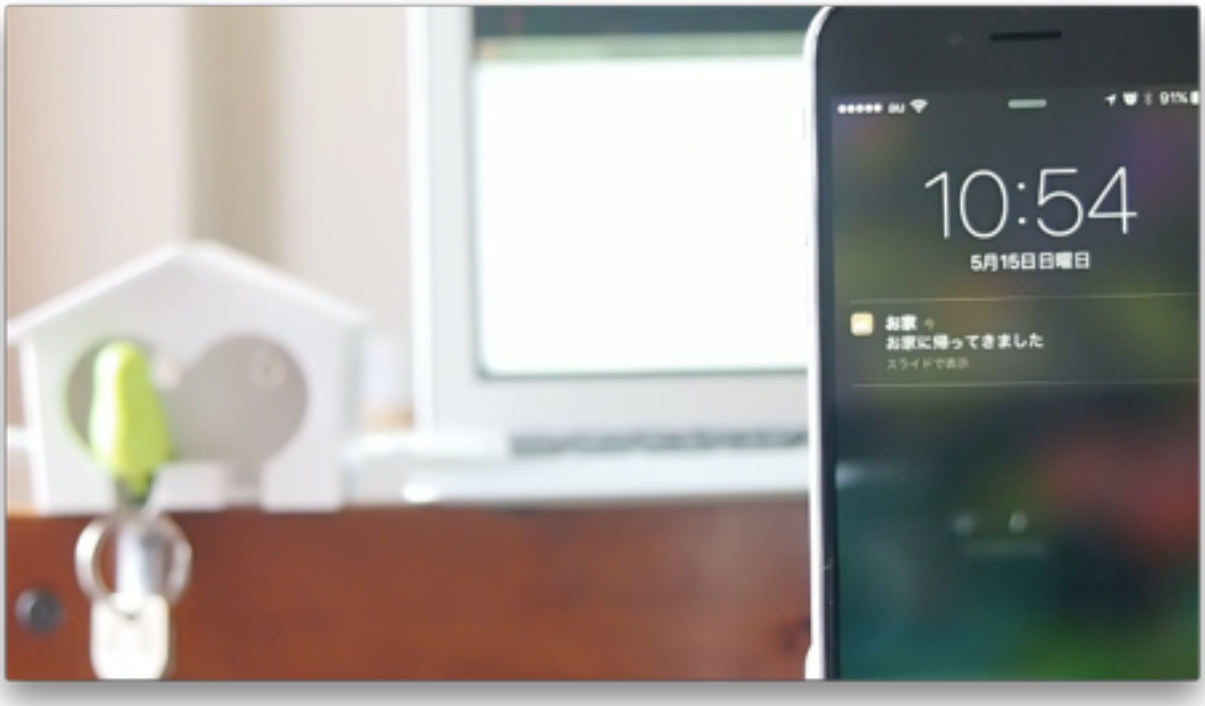
機能の概要を説明します。

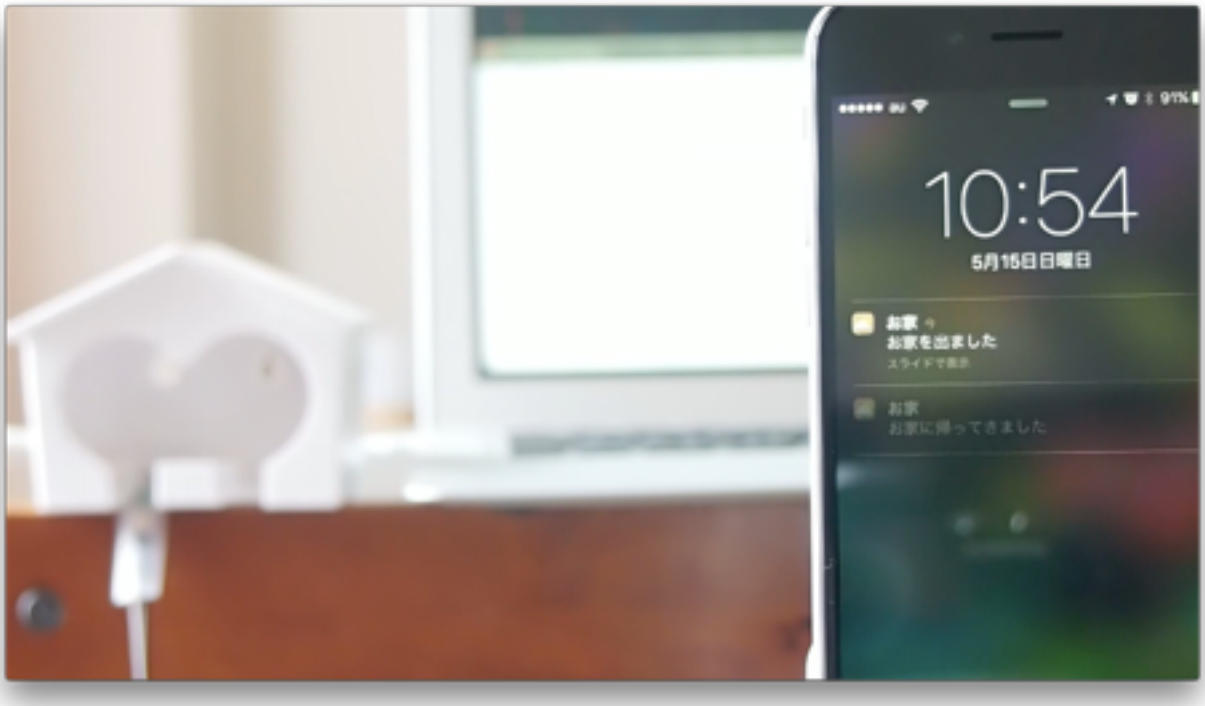
機能

キーボックス付きのキーホルダーです。キーホルダーとRaspberry Piを連携させます。



キーボックスから鍵を取ったら「お家に帰ってきました」、 鍵を戻したら「お家を出ました」というメッセージを、家族のスマートフォンにインストールしたアプリに対して送信します。





制限

今回は簡単のために、家族一人の外出/帰宅を他のもう一人に通知することのみ行います。家族全員の外出/帰宅を、他の家族全員に通知することは今回は行いませんが、実現方法については本文章の最後で議論します。

全体構成

全体構成と構成要素

本機能を実現するシステムの全体図と、その構成要素について説明します。

使用するデバイス・サービス	説明
Duo Sparrow Key Ring	Qualy社製の、鳥と巣箱の壁掛け式キーホルダーです。巣箱がキーボックス、鳥がキーホルダーになります。 http://www.qualydesign.com/product/detail.php?i=109
Raspberry Pi 2	
SFE-SEN-09673	小型の感圧センサーです。
MCP3002	感圧センサーのアナログ入力をデジタルに変換するためのモジュールです。
AWS（Amazon Web Service）	Amazon.com により提供されているクラウドコンピューティングサービスです。今回はこの中のAWS SNSを使用します。
AWS Simple Notification Service (SNS)	AWSで提供されている、アプリケーション、エンドユーザー、およびデバイスがクラウドから通知を即座に送受信できるようにするウェブサービスです。 Push通知をアプリへ送信するのに使用します。
iPhone	通知を受け取るためアプリをインストールするiPhoneです。iPadやiPod touchでも実装可能です。
Mac	今回はMacで開発することを前提に説明します。

必要なアカウント登録について

次のサービスについて事前にアカウント登録をしておきましょう。

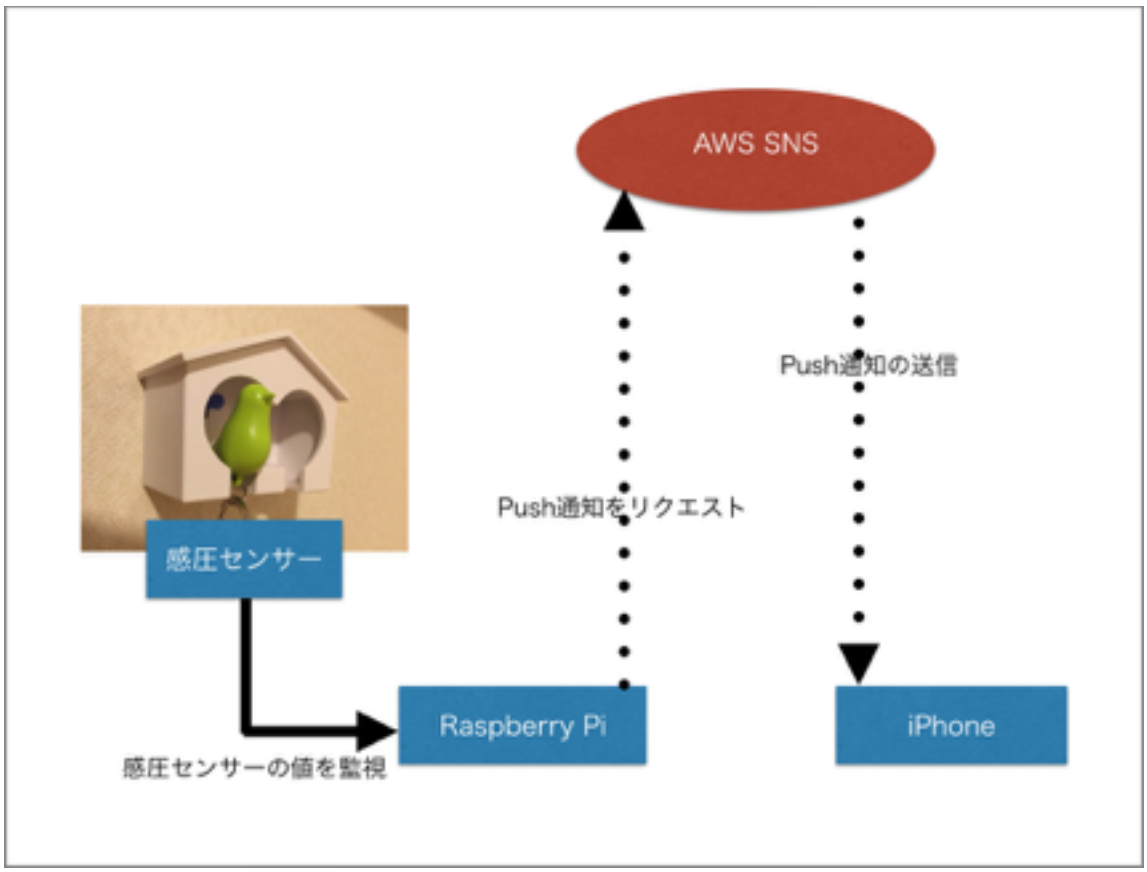
Apple Developer Program

Push通知を受け取るiOSアプリを実装するために必要です。
https://developer.apple.com/

AWS

Push通知を送るためにAWSを使用しますので、アカウントを作成する必要があります。
https://aws.amazon.com/jp/

全体の構成は上図のようになります。
まずはAWS SNSの設定と、その通知を受け取るアプリの実装を行います。



次にRaspberry Piから感圧センサーの抵抗値を取得し、変化を検知したらPush通知をリクエストするよう実装します。

スーマトフォンアプリの実装

具体的な実装方法を説明します。

Apple Developer Programの設定

AWSを使用してPush通知を受け取るためのアプリを実装します。

プロジェクトの作成

アプリの新規プロジェクトを作成します。
Bundle Identifierを決めてください。

Apple Developer ProgramにBundle Identifierを登録

Push通知を受け取るためには、Apple Developer ProgramにてPush通知用の証明書を発行する必要があります。

まずはアプリのBundle Identifierを登録します。次の項目に注意してアプリを登録してください。
登録ページ: https://developer.apple.com/account/ios/identifier/bundle/create

	設定する内容
App ID Description	任意の名前をつけてください。
App ID Suffix	Explicit App IDにアプリのBundle Identifierを入力します。
App Services	「Push Notifications」にチェックを入れます。

開発機を登録

Push通知を受け取る開発機をApple Developer Programに登録します。
登録ページ: https://developer.apple.com/account/ios/device/iphone/create

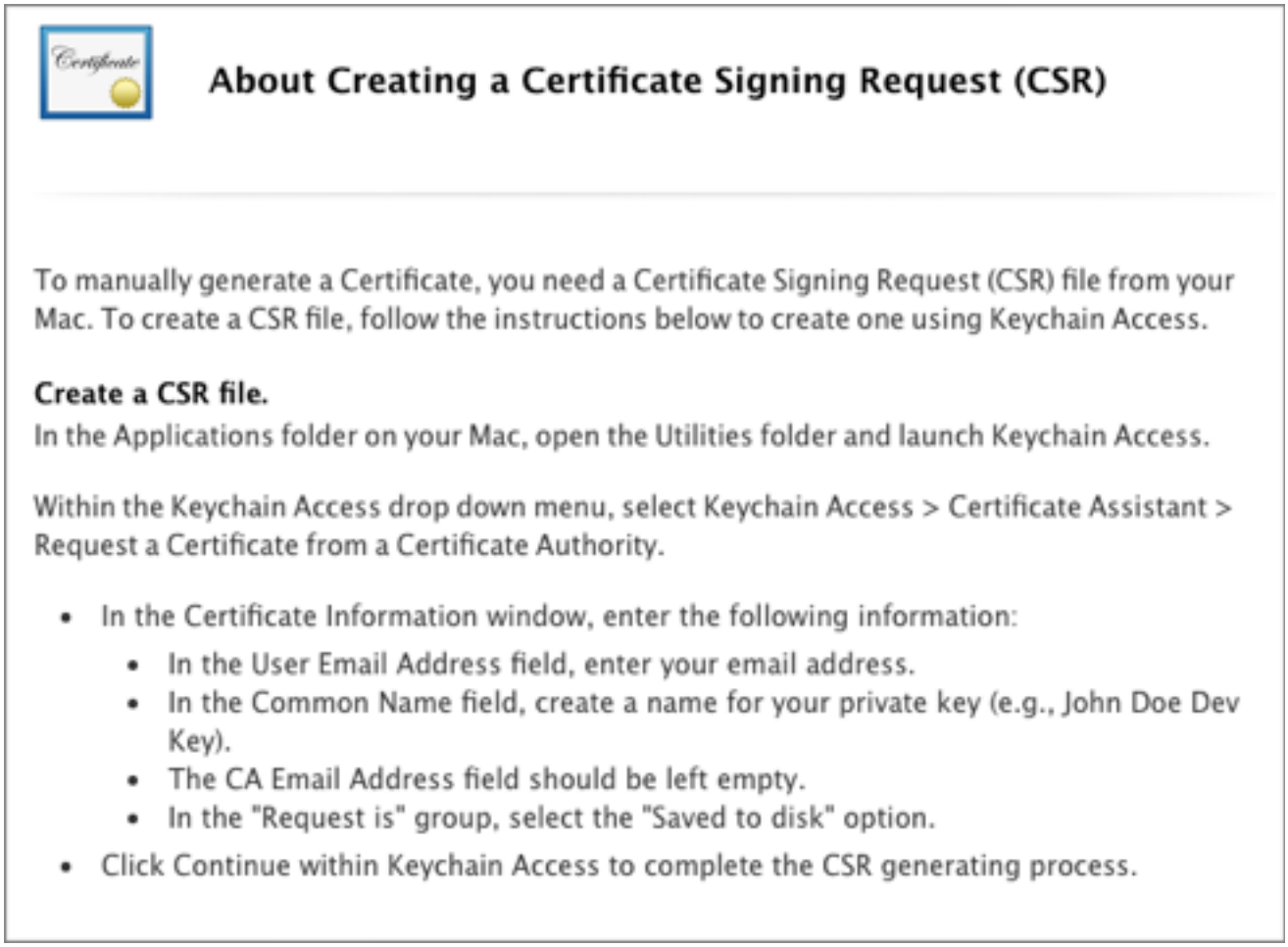
	設定する内容
Name	任意の名前で良いですが、判別しやすい名前をつけてください。
UDID	iPhoneのUDIDです。iTunesに接続することで確認できます。

Push通知用の証明書を作成

次にAWSからPush通知を送信できるよう、証明書を作成します。
証明書は本番アプリ用と開発用がありますが、今回は開発用で作成します。
作成ページ: https://developer.apple.com/account/ios/certificate/development/create

	設定する内容
What type of certificate do you need?	「Apple Push Notification service SSL (Sandbox)」にチェックを入れます。
Which App ID would you like to use?	先ほど登録したアプリを選択してください。

途中、Certificate Signing Requestを求められるので、指示に従って作成してアップロードしてください。



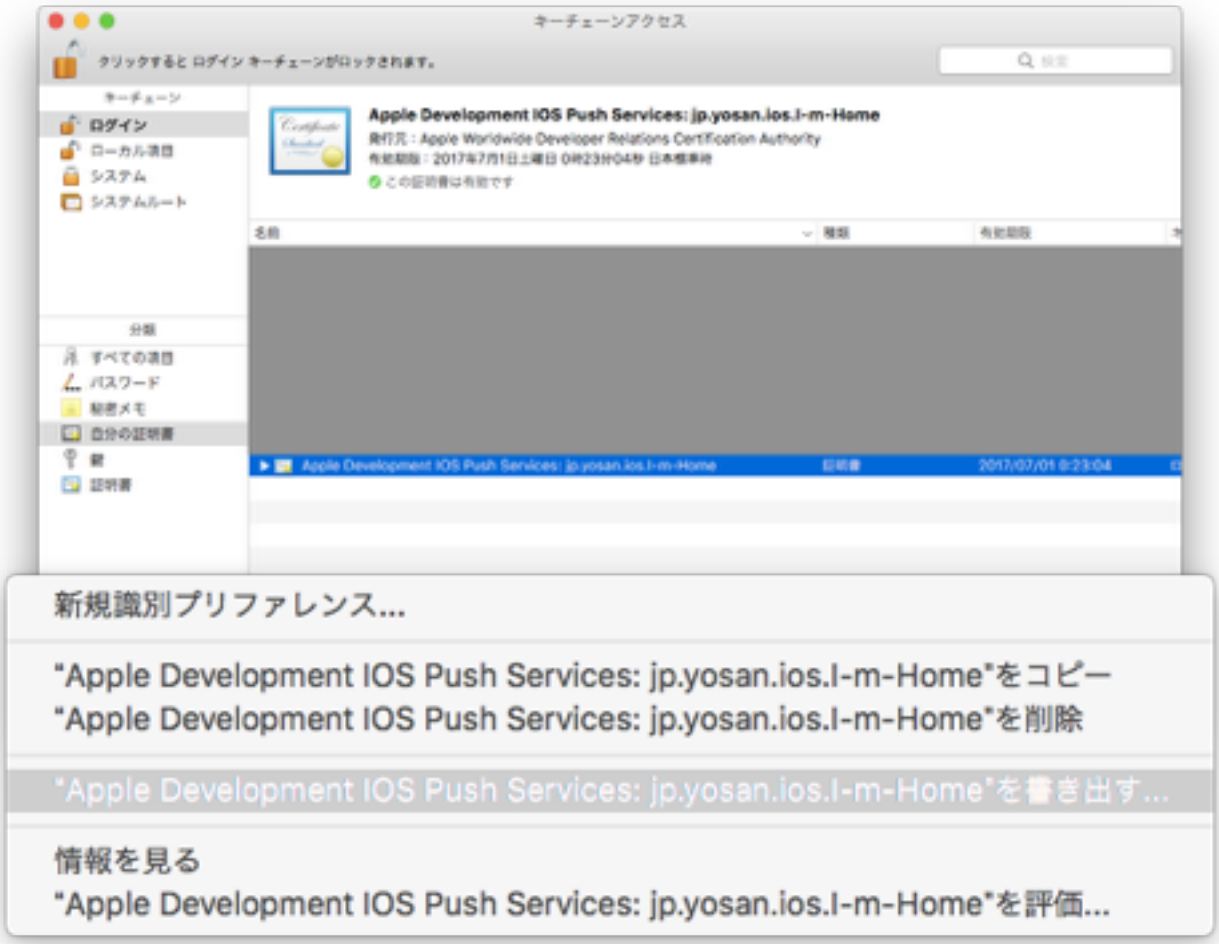
About Creating a Certificate Signing Request (CSR)

To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access.

Create a CSR file.
In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access.

Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority.

- In the Certificate Information window, enter the following information:
  - In the User Email Address field, enter your email address.
  - In the Common Name field, create a name for your private key (e.g., John Doe Dev Key).
  - The CA Email Address field should be left empty.
  - In the "Request is" group, select the "Saved to disk" option.
- Click Continue within Keychain Access to complete the CSR generating process.



作成できたらダウンロードしてファイルをダブルクリックし、Macのキーチェーンアクセスに登録します。登録できたら、証明書をp12ファイルとして書き出します。キーチェーンアクセスで先ほど登録した証明書を選択し、右クリックのメニューから「書き出す」を選択して書き出してください。書き出したファイルは、後ほどAWSにアップロードします。

プロビジョニングプロファイルの作成

アプリがPush通知を受け取るためのプロビジョニングプロファイルを作成します。
作成ページ: https://developer.apple.com/account/ios/profile/limited/create

	設定する内容
Development	「iOS App Development」を選択します。
Select App ID.	先ほど登録したアプリを選択してください。
Select certificates.	自分の証明書を選択してください。
Select devices.	登録したiPhoneを選択してください。
Profile Name	任意の判別しやすい名前を入力してください。

作成したプロビジョニングプロファイルをダウンロードしてダブルクリックすると、Xcodeで使用できるようになります。

アプリにプロビジョニングプロファイルを登録

再度Xcodeを開いて、アプリにプロビジョニングプロファイルを指定します。
プロジェクトファイルを選択し、Build SettingsタブでProvisioining Profileを選択し、先ほど作成したプロビジョニングプロファイル選択します。

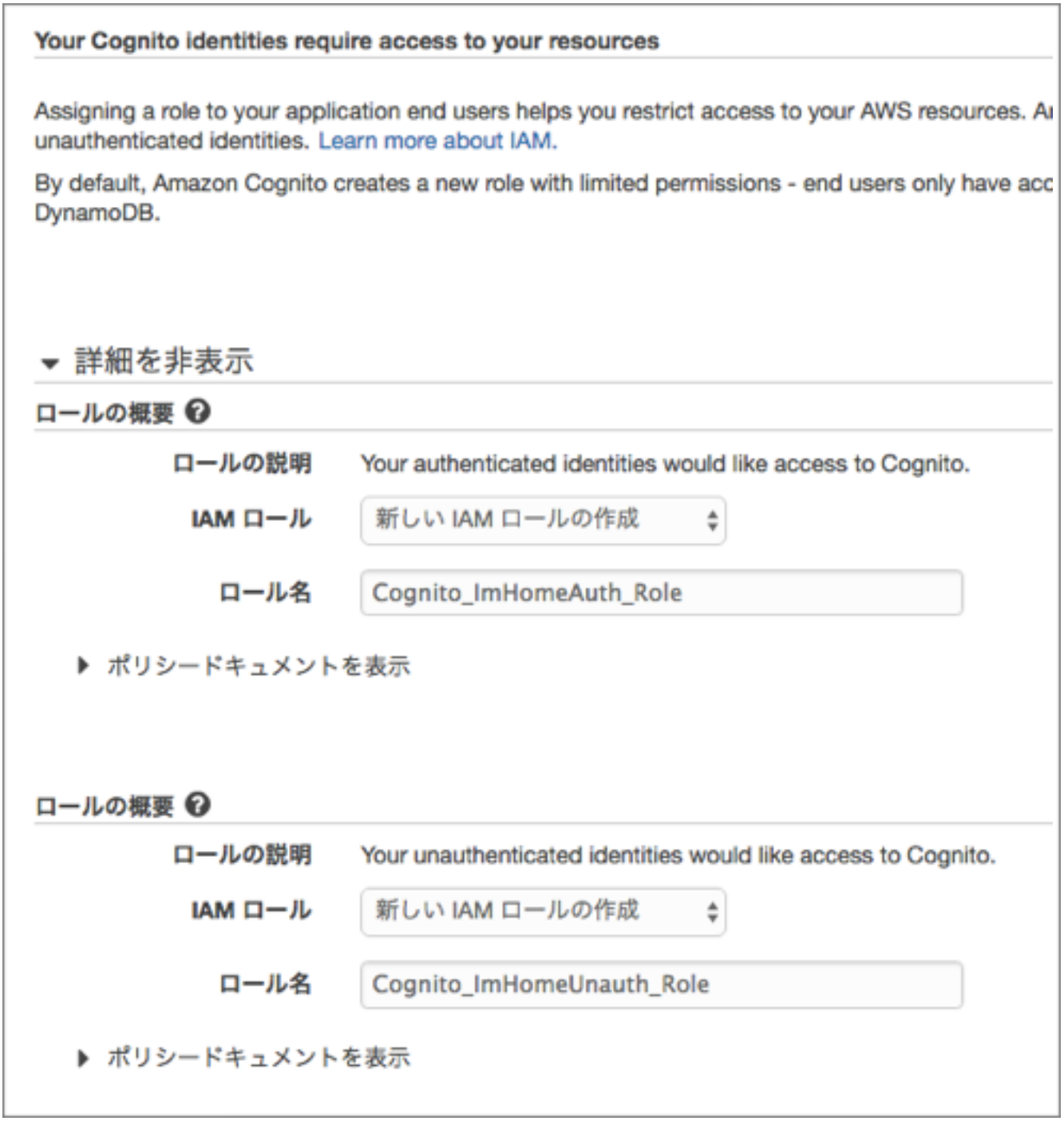
AWSの設定

Push通知を送るための設定をします。

AWS Cognitoの設定

アプリのIdentity Poolを作成する。
作成ページ: https://ap-northeast-1.console.aws.amazon.com/cognito/create?region=ap-northeast-1

	設定する内容
Identity pool name	任意の判別しやすい名前を入力してください。
Enable access to unauthenticated Identities	チェックを入れてください。



途中、ロールを選択する画面になります。今回は簡単のために専用のロールを作成します。
デフォルトで「新しい IAM ロールの作成」が選択されていますので、そのまま進んでください。

この結果作られるIdentityPoolの「**IdentityPoolId**」をメモしておいてください。

AWS SNSの設定

AWS SNSにアプリケーションを登録します。
アプリケーション一覧ページ: https://ap-northeast-1.console.aws.amazon.com/sns/v2/home?region=ap-northeast-1#/applications

「Create platform application」からアプリを登録できます。

	設定する内容
Application name	任意の判別しやすい名前を入力してください。
Push notification platform	Apple development
Push certificate type	iOS push certificate
Choose P12 file	先の行程でキーチェーンアクセスから書き出したp12ファイルを指定します。
Enter password	p12ファイルを書き出すときに指定したパスワードを入力します。

作成後、再度アプリケーション一覧ページに戻ると、作成したアプリケーションが表に表示されています。ここから、対応する**ARN**（arn:aws:sns:ap-northeast-1:*:app/APNS_SANDBOX/*）をメモしておいてください。

IAMの設定

AWS Cognitoで作成したロールにSNSの通知を受け取るための権限を指定します。

ロールの一覧ページ: https://console.aws.amazon.com/iam/home?region=ap-northeast-1#roles

AWS Cognitoの設定で作成したロールでUnauthのものを選択します。
「ロールポリシーの作成」で、次のJSONでポリシーを作成して設定します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
           "Effect": "Allow",
            "Action": [
                "sns:CreatePlatformEndpoint"
            ],
            "Resource": [
                <メモしたARN>

            ]
        }
    ]
}
```

AWS SDKのインストール

AWSはiOS向けのSDKを公開しています。
これを使用することで簡単にAWS SNSのAPIを利用できます。

事前にCocoapodsというiOSのライブラリ依存管理ソフトをインストールしておきます。
ターミナルでアプリプロジェクトのルートフォルダに移動し、「pod init」コマンドを使って
Podfileを作成します。

次にPodfileを編集して、AWSCognitoとAWSSNSを指定します。
下記の例を参考にAWSCognitoとAWSSNSを追加してください。

```
# Uncomment this line to define a global platform for your project
# platform :ios, '9.0'

target 'I-m-Home' do
  # Comment this line if you're not using Swift and don't want to use
dynamic frameworks
  use_frameworks!

  # Pods for I-m-Home
  pod 'AWSCognito'
  pod 'AWSSNS'

end
```

編集できたら、同じフォルダ下で「pod install」コマンドでSDKをインストールしてください。

プログラムの作成

通知を受け取るために、次の処理をアプリ起動時に実行しています。
1. デバイストークンをAPNSから受け取ります。
2. AWS SNSへデバイストークンを登録し、Endpoint Arnを発行してもらいます。
3. Endpoint Arnをログに出力します。

上記コマンドの後に作成される、xcworkspaceを開いてください。
今回は簡単のために通知を受け取るプログラムのみ記入します。

```
import UIKit
import AWSCognito
import AWSSNS

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    private let identityPoolId = <AWS Cognitoで作成したIdentityPoolのID>
    private let platformApplicationArn = <AWS SNSで作成したアプリのARN>


    func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
        let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .APNortheast1,
identityPoolId: identityPoolId)
        let configuration = AWSServiceConfiguration(region: .APNortheast1,
credentialsProvider: credentialsProvider)
        AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
configuration
        let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound,],
categories: nil)
        UIApplication.sharedApplication().registerUserNotificationSettings(settings)

        return true
    }

    func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
        application.registerForRemoteNotifications()
    }

    func application(application: UIApplication,
didRegisterForRemoteNotificationsWithDeviceToken deviceToken: NSData) {
        let deviceTokenString = "\(deviceToken)"
            .stringByTrimmingCharactersInSet(NSCharacterSet(charactersInString:"<>"))
            .stringByReplacingOccurrencesOfString(" ", withString: "")
        let request = AWSSNSCreatePlatformEndpointInput()
        request.token = deviceTokenString
        request.platformApplicationArn = platformApplicationArn
        AWSSNS.defaultSNS().createPlatformEndpoint(request)
            .continueWithSuccessBlock { (task) -> AnyObject? in
                guard let response = task.result else { return nil }
                print("Your Endpoint Arn: \(response.endpointArn)")
                return nil
            }
            .continueWithExecutor(AWSExecutor.mainThreadExecutor(), withBlock: { (task) ->
AnyObject? in
                if let error = task.error {
                    print("error: \(error.code), \(error.description)")
                }
                return nil
            })
    }

    func application(application: UIApplication,
didFailToRegisterForRemoteNotificationsWithError error: NSError) {
        print("error: \(error.code), \(error.description)")
    }

    func application(application: UIApplication, didReceiveRemoteNotification userInfo:
[NSObject : AnyObject]) {
        print(userInfo)
    }

}
```

Push通知の確認

Push通知の動作確認をします。

1. 上記のアプリをビルドしてiPhoneにインストールし一度起動したらホームボタンで閉じます。
2. AWS SNSで作成したアプリを選択します。
3. Endpoint ARNにエントリーがあることを確認します。
4. Endpoint ARNを選択して「Publish to endpoint」を選択し、メッセージ送信用のJSONを設定してpublishします。

```
{
    "APNS_SANDBOX":"{\"aps\":{\"alert\":\"test\"}}"
}
```

5. アプリに通知が届いたら設定完了です。

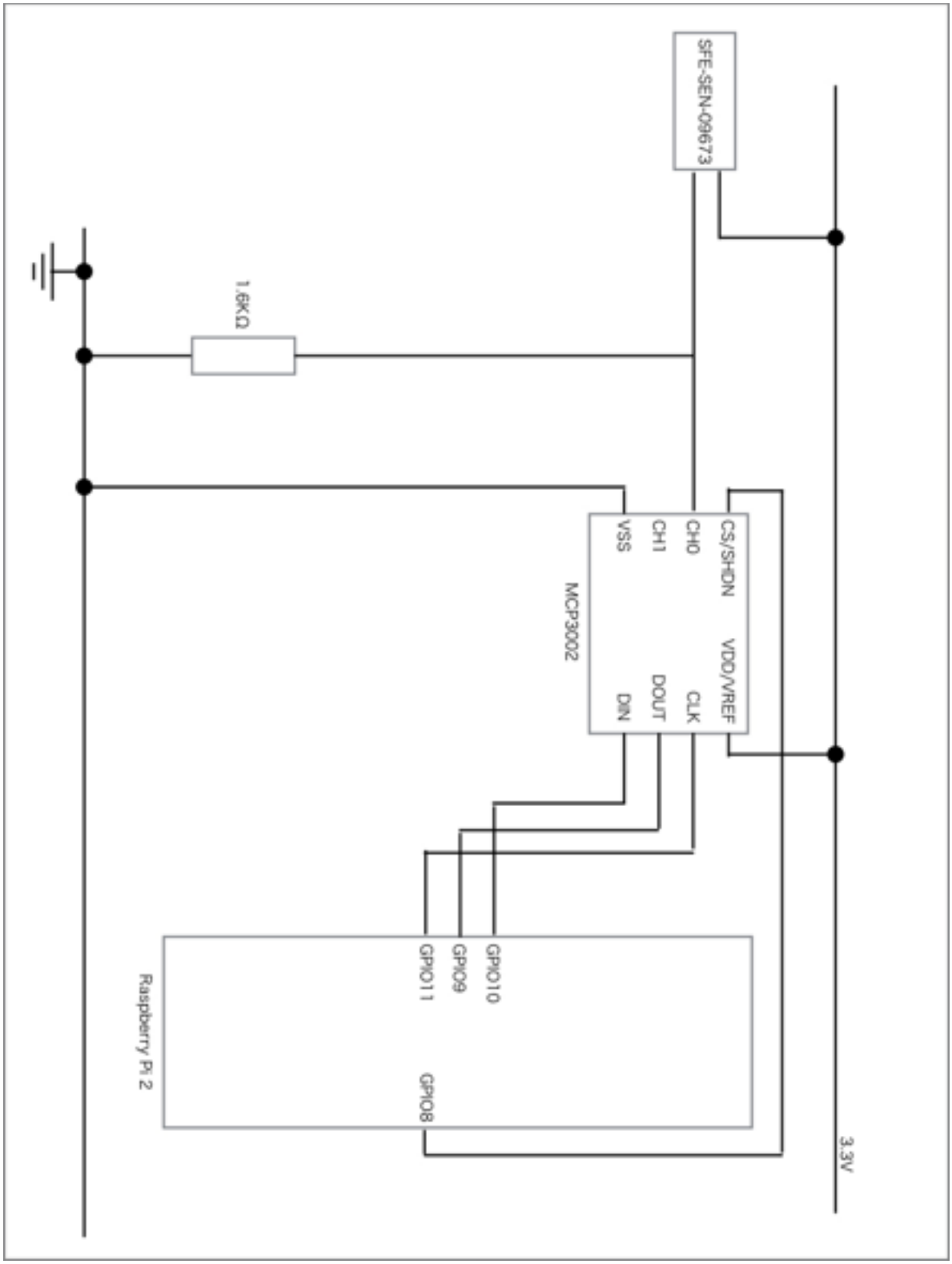
このときの**Endpoint ARN**をメモしておいてください。

以上でアプリの作成は完了です。

Raspberry Piの設定

配線

図を参考に配線してください。



感圧センサー（SFE-SEN-09673）の抵抗値はアナログ値として取得されるため、アナログ入力をMCP3002でデジタル出力に変換し、Raspberry Pi 2に入力するようにします。

感圧センサーをキーホルダーへ取り付け

キーホルダーの鳥の尾が触れる位置に感圧センサーを取り付けます。動作を確認しながら調整してください。

AWS IAMの設定

Raspberry PiがPush通知を送れるようIAMの設定を行います。
ユーザー一覧画面: https://console.aws.amazon.com/iam/home?region=ap-northeast-1#users

1. 「新規ユーザの作成」を選択して任意のユーザ名で作成します。
2. 「認証情報のダウンロード」をクリックして「Access Key Id」と「Secret Access Key」を含むcsvファイルをダウンロードします。
3. 「ポリシー」→「ポリシーの作成」から次のJSON設定を持つポリシーを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1467648561000",
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource":"arn:aws:sns:*"
        }
    ]
}
```

4. 「ユーザー」から先ほど作成したユーザーを選択し、「ポリシーのアタッチ」で先ほど作成したポリシーをアタッチします。
5. Raspberry PiにAWS CLIをインストールします。

```
$ sudo pip install awscli
```

6. 「aws configure」コマンドで、先ほどダウンロードした「Access Key Id」と「Secret Access Key」を入力してください。Regionは「ap-northeast-1」です。

プログラム

下記のpythonプログラムを書いてください。誤動作を防ぐために、キーホルダーを置いた後/取った後でしばらくしたら通知が送信されるようにしています。
先にspiの有効化とboto3のインストールが必要です。

```
$ sudo pip install boto3
$ sudo vi /boot/config.txt
dtparam=spi=on
$ sudo reboot
```

```
# coding:utf-8
# Read the analog sensor value via MCP3002.

import spidev
import time
import subprocess
import urllib2
import boto3

# input
target_arn = <AWS SNSでメモしたEndpoint Arn>


# sns
sns = boto3.resource('sns')
platform_endpoint = sns.PlatformEndpoint('arn')

# open SPI device 0.0
spi = spidev.SpiDev()
spi.open(0, 0)

# parameters
count = 0
athome = False

try:
    while True:
        resp = spi.xfer2([0x68, 0x00])
        value = (resp[0] * 256 + resp[1]) & 0x3ff
        print 'value: {0:2d} count: {1:2d}'.format(value, count)

      if not athome and value > 0:
          if count == 5:
            result = platform_endpoint.publish(
                     TargetArn=target_arn,
                     Message='お家に帰ってきました'

                )
            print(result)
            athome = True
          else:
            count += 1
      elif athome and value == 0:
          if count == 5:
            result = platform_endpoint.publish(
                     TargetArn=target_arn,
                     Message='お家を出ました'

                )
            print(result)
            athome = False
          else:
            count += 1
      else:
          count = 0

        time.sleep(1)
except KeyboardInterrupt:
    spi.close()
```

動作確認

pythonプログラムを実行し、キーホルダーを取ったり外したりして、通知がiPhoneに送信されれば成功です。うまく送信されない場合は感圧センサーの位置を調整してください。

実用化に向けての課題

感圧センサーを使うことに関する課題

感圧センサーの課題として、鍵の置き方や重さによって反応が悪いことが挙げられます。また、今回は家族の誰かが、他の誰か一人に通知を行う機能を実装しましたが、実際はお互いの帰宅や外出を通知することが求められます。

電源に関する課題

今回の実装ではRaspberry Piを使用しましたが、常にセンサーを監視しつつWiFi接続を常時行う場合では電力消費が大きくなります。一方で玄関には電源がないことも多いため、より省電力で動作する仕組みが必要になります。