MIT App Inventor Particle Photon Test
Installation and User Manual

Version 1.1; date: 5/07/2018

NOTICE: Use of this document is subject to the terms of use described in the document "Terms_of_Use_License_and_Disclaimer" that is included in this release package. This document can also be found at:
https://github.com/TeamPracticalProjects/MIT-App-Inventor-Particle-Photon-test/blob/master/Terms_of_Use_License_and_Disclaimer.pdf

Table of Contents

Table of Contents

1. Introduction.

This project demonstrates how a maker, hobbyist or prototype developer can use the MIT App Inventor 2 (AI2) Integrated Development Environment (IDE) to create apps that can communicate, over the Internet, with Particle Photon and Electron devices. Specifically, an example app is provided (source code as well as a compiled, executable version) that shows how to use MIT App Inventor 2 code to read variables from a Particle device and to call functions and receive function return values from a Particle device, over the Internet.

Particle (particle.io) is a company[1] that produces Internet of Things (IoT) microcontroller modules. Two of their current products are of interest to makers, hobbyists and prototype developers:

Photon: An inexpensive ($19) module with an advanced, 32 bit microcontroller and built-in WiFi communication capability.

Electron: Similar module to a Photon, but with 2G/3G cellular Internet capability, including the option for an inexpensive cellular data plan.

All Particle devices communicate over the Internet using *Particle Cloud* software. Particle hosts a cloud service that can be used, at no charge, by their maker, hobbyist, and commercial prototype customers.

Particle devices are programmed using an Arduino-like programming language, with extensions for cloud communication:

> *Particle.variable():* Exposes a variable defined within the Particle microcontroller program (firmware) to the Particle Cloud.
>
> *Particle.function():* Calls a function defined within the Particle microcontroller program (firmware) from the Particle Cloud.
>
> *Particle.publish():* Publishes events defined within the Particle microcontroller program (firmware) to the Particle Cloud.
>
> *Particle.subscribe():* Subscribes to events defined within the Particle microcontroller program (firmware) from the Particle Cloud.

This project demonstrates building an app that can:

(a) Read variables from a Particle device into an app created using AI2.

[1] Team Practical Projects has no relationship to Particle. We are simply users of their products.

(b) Call functions and process function return values on a Particle device using an app created in AI2.

This demonstration app is integrated with our *Particle App Template* (https://github.com/TeamPracticalProjects/Particle_App_Template). The Particle App Template (*Template*) provides the app user with a means to log in to their Particle account and select a Particle device registered to their account to be used with the app. The Template allows apps created in this manner to be built and distributed without the need for the user to hardcode their Particle *user access_token* and *device_ID* into the app.

This document provides instructions for installing and using this demonstration app. An accompanying document describes the source code used to read Particle variables and call Particle functions, as a guide for app developers:

https://github.com/TeamPracticalProjects/MIT-App-Inventor-Particle-Photon-test/blob/master/Documentation/Programming_Notes.pdf

2. Installation.

In order to demonstrate how this app works, it is necessary to build a project with a Particle device, some external hardware, load demonstration firmware on to the Particle device, and then to install the app on a smartphone or tablet[2]. This section provides instructions for building and installing hardware and software/firmware to run this demonstration. The next section of this document provides detailed instructions for using the app with the Particle device hardware and firmware. For the remainder of this document, we will assume that the Particle device is a Photon; however, the same instructions apply to the Particle Electron.

2.1. Hardware.

The hardware for this project consists of the following:

- Particle Photon (or Electron).
- Normally open, momentary pushbutton switch.
- LED and current limiting resistor
- 0.1 uF decoupling capacitor
- Hobby servo

There are two ways to build this hardware:

[2] MIT App Inventor 2 currently supports only Android smartphones and tablets. MIT has promised support for iOS later in 2018. There are also commercially available Android emulators for OSX and Windows that allow apps created in AI2 to be run on these platforms.

(1) On a solderless breadboard.
(2) Using the printed circuit board of our Water Leak Sensor project:

https://github.com/TeamPracticalProjects/WaterLeakSensor

2.1.1. Using the Water Leak Sensor Project Hardware.

You can use some of the hardware from our Water Leak Sensor for this project. See: https://github.com/TeamPracticalProjects/WaterLeakSensor for details. Using the hardware from this project provides you with the advantages of a printed circuit board to mount components neatly and 3.3 volt to 5 volt conversion for driving the LED and Servo motor. Furthermore, you can build the full Water Leak Sensor when you are done and use it to monitor temperature and humidity at some location and send you alarms whenever a water leak is detected!

You will not need all of the components of the Water Leak Sensor hardware for the demonstration purposes of this project. Specifically:

(1) You can leave the DHT11 temperature/humidity sensor and the Piezo Buzzer off of the printed circuit board. These are not used for this demonstration project. Otherwise, fully assemble the printed circuit board with all components indicated.

(2) The following are the only external components that you will need to connect to the printed circuit board for this demonstration project: Backlit pushbutton switch and 5 volt hobby servo. These are wired up as indicated in the Water Leak Sensor project documentation. You will not need the enclosure or the bezel for this demonstration project, nor will you need the toggle switch, water level sensors, nor their intermediate connectors and cabling.

2.1.2. Using a Solderless Breadboard.

You can mount the Photon on a solderless breadboard and power the entire project using the micro-USB connector on the Photon. Connect the 3.3v pin on the Photon to the positive supply voltage rail of solderless breadboard and one of the GND pins on the Photon to the ground (negative supply) rail of the solderless breadboard.

You will need the following external components:

(1) A normally open, momentary pushbutton switch[3]. You can get miniature switches that mount directly on a solderless breadboard if you wish (e.g. https://www.mouser.com/ProductDetail/ALPS/SKHHAKA010/?qs=seHrhfPpLDyPUsCTaeDTPQ%3D%3D&gclid=Cj0KCQjw5qrXBRC3ARIsAJq3bwrEqTMn4Wp5qFhpkbKca8os

[3] This switch is not actually used with the current version of the app. It is included in the hardware and Particle firmware for possible future development of push notifications using Particle.publish().

TdHO2L5YbLSr-pMxPGu9Hrkvi0TlD7MaAko9EALw_wcB). Connect one side of the switch to the GND (negative supply) power rail of the solderless breadboard and the other end of the switch to Photon pin D4.

(2) An LED (3 mm or 5 mm, any color) and a current limiting resistor. For driving the LED from 3.3 volts, a current limiting resistor anywhere between 180 ohms and 220 ohms (at least 1/8 watt) will be sufficient. Connect Photon pin D5 to one end of the resistor. Connect the other end of the resistor to the anode (long lead) of the LED. Connect the cathode (short lead) of the LED to GND (negative supply).

(3) A 3.3 volt compatible servo[4]. Connect the +3.3 volt supply lead of the servo to the positive supply rail on the solderless breadboard and wire the ground pin of the servo to GND (negative supply rail) of the breadboard. Connect Photon pin A5 to the servo control lead.

(4) Connect a 0.1 microfarad capacitor between the positive supply rail and the GND (negative supply rail) of the solderless breadboard near where the servo connects to these supply rails. This will decouple electrical noise generated by the servo motor from the supplied power.

2.2. Firmware Installation and Test.

In order to provide something for the demonstration app to communicate with, you will have to install ("flash") the firmware "Test_MIT.ino" onto your Photon. This firmware is released in source code form and can be found at:

https://github.com/TeamPracticalProjects/MIT-App-Inventor-Particle-Photon-test/blob/master/Particle_Firmware/src/Test_MIT.ino

Download this file to your computer and then use the Particle Web IDE to compile it and flash it to your Photon (or Electron).

If you are new to Particle, go to the Particle's on-line documentation and read the instructions for getting started, including setting up your Particle account, downloading the Particle app to your smartphone to configure and capture your Photon to your Particle account, getting your Photon on-line to the Particle cloud (running Tinker) and using the Particle Web IDE to compile source code and flash it to your Photon device. Details can be found at:

https://docs.particle.io/guide/getting-started/intro/photon/

[4] Many standard 5 volt hobby servos will also work with the control from a 3.3 volt pin. Connect the +5 volt supply lead of the servo to the VIN pin of the Photon and wire the ground pin to GND (negative supply rail) of the breadboard. Connect Photon pin A5 to the servo control lead. Connect the decoupling capacitor between the VIN pin and GND near where the servo power supply leads are connected.

Once your Photon is up and running and you can communicate with it via the Tinker app, read the documentation about the Particle Web IDE on this same web page. Then head over to the Particle Web IDE (https://build.particle.io/build/new), log in to your Particle account, start a new project by typing the name of the firmware into "Current App", and copy/paste the source code from the "Test_MIT.ino" file that you downloaded to your computer into the Web IDE code window. *Make sure that you overwrite the default code that is in the code window*:

```
void setup() {

}

void loop() {

}
```

since the source code provided in "Test_MIT.ino" contains setup() and loop() functions.

Make sure that you have copied the entire contents of the file "Test_MIT.ino" into the source code window of the Particle Web IDE and click on *save* icon (the little folder icon) to store it on the Particle server. Next click the *compile* icon (check mark in a circle) and make sure that the code compiles OK. Finally, check that your Photon is the selected device and click on the *flash* icon (lightning bolt) on the Web IDE to flash the code to your Photon. Once the code is flashed successfully, your Photon should be "breathing cyan" and the D7 LED on the Photon should be flashing at a 1 second rate.

You should next use the Particle Console to test that your Photon with the "Test_MIT.ino" firmware is working properly. Click on the *Console* icon at the bottom left of the Web IDE (the little bar graph) to open up your Particle Console window in your web browser.

Once in the Particle Console window, click on the *My Devices* icon (the little cube icon) to see a list of your Particle devices. Select the device that you have flashed the "Test_MIT.ino" firmware to. You should now see a screen that looks like figure 2.1:

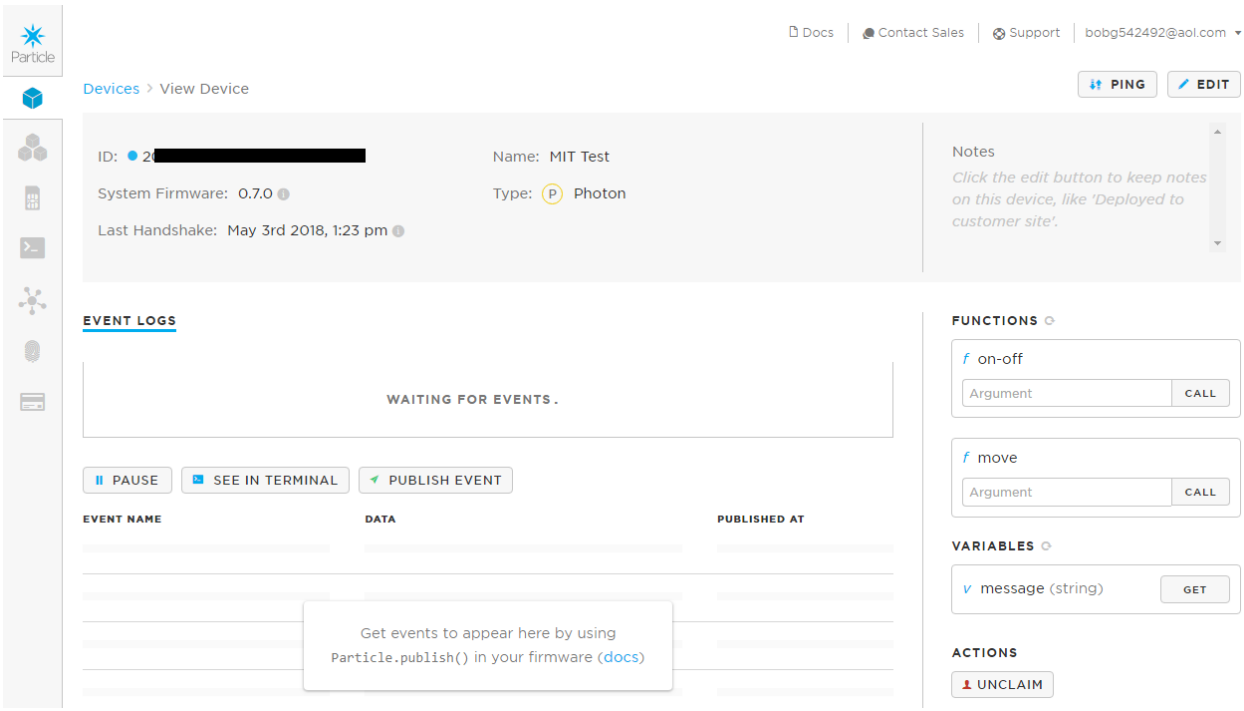

Figure 2.1. Particle Console for Selected Device.

Note the FUNCTIONS and VARIABLES listed on the right hand side of this screen. You can use these to test your device.

Click the GET button in the variable called “message”. The Console will retrieve the message from your Photon device and display it here. The message will look something like:

v message (string) = **firmware version 1.0; last reset at:Sun Apr 15 17:39:28 2018Z**

Next, look at the FUNCTION “on-off”. Type “on” in the Argument field and click CALL. The LED attached to pin D5 of your Photon should light up and the function return value should be “1” on the Console. Now type “off” in this same field and the LED should turn off and the function return value should show as “0”.

Finally, look at the FUNCTION “move”. Type a servo angle into the Argument field (a number between 5 and 175) and click CALL. The servo should move to that angle and the function return value should show the same angle as you typed in[5]. You have now successfully tested the hardware and firmware for this project.

[5] If you put in a number that is less than 5, the servo will move to 5 degrees and the function return value will be 5. Likewise if you put in a number greater than 175, the servo will move to 175 degrees and the function return value will be 175. If you type in something that is not a number, the function return value will be -1 and the servo will not move.

2.3. App Installation.

Download the file "CommunicateWithParticleDemoApp.apk" to your computer from:

https://github.com/TeamPracticalProjects/MIT-App-Inventor-Particle-Photon-test/blob/master/Software_MIT_App_Inventor/Deployable%20App/compiled/CommunicateWithParticleDemoApp.apk

Next, transfer this file to your Android smartphone or tablet. You can transfer the file either by USB cable or by e-mailing it to yourself as an attachment. Either way, store the file "CommunicateWithParticleDemoApp.apk" in a location that you can find on your smartphone or tablet using the "My Files" app on your Android device[6].

Use "My Files" to locate this file and tap on it to install the app on your device. **Note**: your device might pop up a window warning you about installing an app from an unknown source. If this happens, tell Android to go ahead and install it anyway[7]. After installation, you should be able to locate and open the app like any other app on your smartphone or tablet and use it to operate your demonstration hardware. Operational instructions are provided in the next section of this document.

3. Operation.

This section describes the operation of the demonstration app and how it interacts with the Particle device that you built and tested, as described in section 2. The app is integrated with the Particle App Template so that in addition to the app performing the basic get Particle cloud variable data and call Particle cloud functions, the app provides the user with a means to log in to their Particle account and select the Particle device to use. See the following for more information about the Particle App Template:

https://github.com/TeamPracticalProjects/Particle_App_Template

When you first open the app, it should look similar to figure 3.1. Note the yellow warning message at the bottom right hand corner of the opening screen. This appears after first installation of the app because no Particle device has been selected. Tap the *Setup* button at the lower left hand corner of the screen to go to the setup screen (provided by the Template).

[6] If you have an SD card in your device, we suggest storing the .apk file to the root directory of the SD card, i.e. /SDcard in the device's file system.

[7] The exact procedure is Android version specific, but it will be obvious how to proceed to install the app from here.

Figure 3.1. Initial Screen Display After Installing the App.

After tapping *Setup*, you will be on the setup screen, similar to figure 3.2. Note in figure 3.2 that the *Particle User ID* and *Particle User PW* fields are empty. The first time that you use the app, you must enter your Particle User Name (Login) and Particle PW (Password). Also note the button with the label *No Devices.* The app (Template) has not received a list of your Particle devices from the Particle Cloud until you successfully log in.

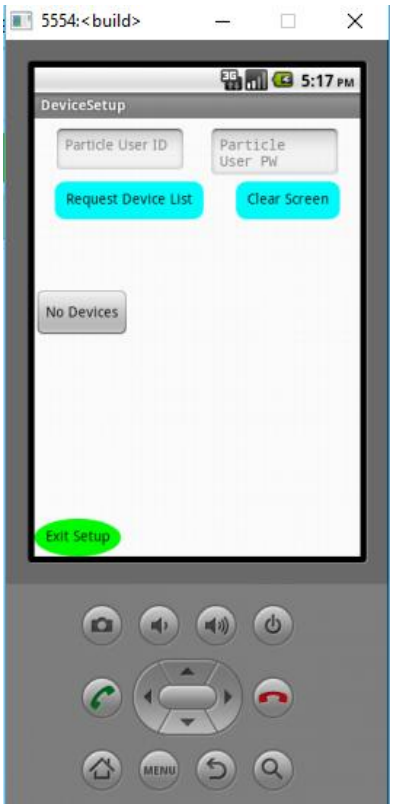

Figure 3.2 Initial Setup Screen.

After entering your Particle User ID and Password, tap *Request Device List*. If your User ID and Password are correct, you have successfully logged into the Particle Cloud and the Particle Cloud has returned a list of Particle devices that are registered to your account. Figure 3.3 is representative of what the app setup screen should look like at this point in the process.

Figure 3.3. Login Successful, Device List Obtained from Particle Cloud.

Referring to figure 3.3, your Particle User ID should be displayed along with an “OK” response and an http response code in the 200 range. This indicates successful login to the Particle Cloud. You will also notice that the *No Devices* button label has changed to *Select Device*, indicating that the app (Template) has received a list of devices that are registered to your Particle account. Tap on *Select Device*. A pick list of devices in your Particle account will appear, similar to figure 3.4.

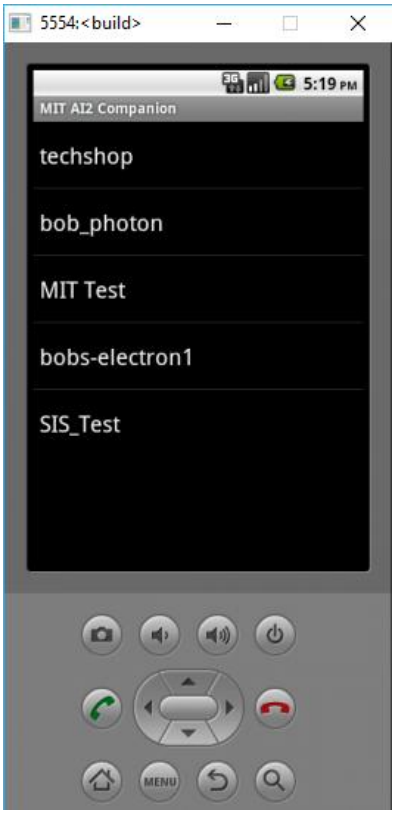

Figure 3.4 Device Pick List.

Select the device that you are using for this project by tapping on an entry in the list in figure 3.4. The app (Template) will now query the Particle Cloud for the selected Particle device and return the results, as indicated in figure 3.5.

Figure 3.5. Device Selected.

Referring to figure 3.5, the selected device *name* and *device_ID* will be displayed on the app, and the user credentials and selected device information will be persistently stored inside the app (Tiny DB). You can now tap *Exit Setup* to return to the main screen of the app, as depicted in figure 3.6.

Figure 3.6 Return to Main App Screen with a Device Selection.

Referring to figure 3.6, the app (Template) retrieves the user and device credentials from the app's internal storage (Tiny DB) and uses these to "ping" the device to ascertain its online status. If your Photon is powered and connected to the Particle Cloud, the status should show as *online* in green, as shown in the lower right hand corner of figure 3.6.

Hereafter, whenever you open the app, it will automatically retrieve the user and device credentials from the internal app storage (Tiny DB) and automatically ping the device for its current online/offline status[8]. The user will not need to go back to the setup screen unless they wish to change device or log in to a different Particle account.

At this point, the Template work is done and the demonstration app code may be executed. The demonstration app code allows the following to be performed on the Photon via the Particle Cloud:

(1) Retrieving the firmware-created device status message from the Particle.variable() declaration.

[8] Depending upon how the app was last exited, the Android OS may cache the main screen and therefore not execute the device ping code. The user can always ping the device, at any time, by tapping *Refresh*.

(2) Turning the LED on and off via the Particle.function() declaration for the “on-off” function in the firmware, and processing the function return value.

(3) Moving the servo via the Particle.function() declaration for the “move” function in the firmware, and processing the function return value.

The demonstration app firmware has been coded to obtain the status message from the device after a successful device ping. This is the variable (String, in this case) that is exposed to the Particle Cloud via Particle.variable() in the firmware. The app displays this message, containing firmware version and last device reset time, in the label that is just above the Template buttons and status near the bottom of the screen; see figure 3.6.

Figure 3.7. Turn the LED ON and receive a Response from the Photon.

Tap the *LED ON* button in the app and the LED turns on and the LED ON button turns green, as shown in figure 3.7. Turning the button green is the result of a “1” response returned from the “on-off” cloud function call to turn the LED on. Likewise, tapping *LED OFF* turns the LED off and the LED ON button returns to no color and the LED OFF button turns green in response to the cloud function return code of “0”, as shown in figure 3.8.

Figure 3.8. Turn the LED Off.

The user can turn the LED on and off repeatedly and observe both the hardware response and the app processing of the cloud function return values via changing the buttons' color between none and green.

The servo can be moved either using the slider control on the app or by typing the desired servo angle into the text box to the right of the slider control on the app. Moving the slider displays the slider angle selection in the text box, as shown in figure 3.9.

Figure 3.9. Select Servo Angle with the Slider.

The user can move the slider around until the proper servo angle command value is displayed in the text box to the right of the slider control, as shown in figure 3.9. To actually move the servo, the user taps *SET SERVO*. The servo moves to the commanded angle and the actual servo command value that the firmware sends to the servo is returned to the app and is displayed to the right of the *SET SERVO* button. In this case, it is the same as the commanded value because the commanded value is within the firmware-dictated range of 5 to 175 degrees.

The servo can also be commanded by typing a value directly into the text field and not using the slider control. This operation is depicted in figure 3.10. Note that the slider does not move in response to changing the value by typing a number – this is how this demonstration code was written and, of course, the code can be written for any desired behavior.

Figure 3.10. Select Servo Angle Via Typing A Number.

Figure 3.10 is an example of typing a valid servo angle into the text box. The servo position returned by the firmware via the Particle Cloud is the same as the commanded value because the firmware was written to accept any servo angle command value between 5 and 175 degrees. Figure 3.11 shows what happens if an invalid servo angle is entered. The firmware is written to clamp any value less than 5 to 5, and any value greater than 175 to 175, and then use this clamped value to command the servo hardware. The returned value, as shown in figure 3.11, is the clamped value and not the commanded value, demonstrating the value of having the app process function return values from the Photon firmware.

Figure 3.11. Typing an Out Of Range Servo Angle.

This completes a walkthrough of the operation of the demonstration app. This project is only a demonstration. The project developer may use the source (firmware and software) code posted with this project as a guide to how to control and status Particle devices using an AI2 developed app and how to integrate their project-specific app code with our app Template.