Afero Developer Training
Lab Handbook

for Afero Modulo-1, Modulo-2, Profile Editor

April 25, 2018 - Version 1.5

Overview

This document supports hands-on training sessions for using the following Afero products, designed to work with the Afero Platform:

- Afero Modulo-1 and Modulo-2 hardware, and the
- Afero Profile Editor developer tool.

Lab 1. Afero Onboarding

This lab walks you through:

- Setting up an account on the Afero Cloud.
- Connecting a Modulo development board to your account.
- Putting the board through its paces.

This lab takes about 30 minutes to complete. Note that the lab illustrates onboarding for a Modulo-2, but the general flow is the same for onboarding a Modulo-1, minus the Wi-Fi setup.

1.1 Prerequisites

There are no prerequisites for this lab.

1.2 Tools/Materials

Before beginning the lab, you will need:

- iOS or Android mobile phone, running:
  - Android KitKat 4.4 (API Level 19), or
  - iOS release 9.3
- Afero Modulo development board
- Micro-USB cable to power the Modulo
- USB power source (free USB port on a computer, AC wall adapter, or portable charger)

1.3 Setup

Download the Afero mobile app for your smartphone. The links below should take you to the appropriate download page for your device, or you can search your app store for “Afero”:

- Android: https://play.google.com/store/apps/details?id=io.afero.tokui.prod.release
- iOS: https://itunes.apple.com/us/app/afero-iot-platform/id1065087421?mt=8

NOTE Users in China should be able to access the app through Google Play because it is a free app. Afero does not distribute the app in any other way because the app stores provide a way to update the app when required.

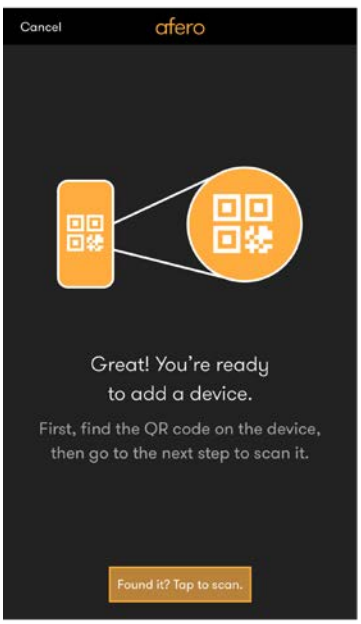
1.4 Steps

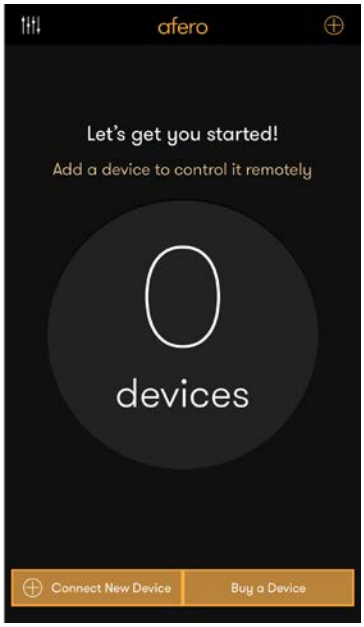
1 Launch the Afero mobile app on your smartphone. To create an account on the Afero Cloud, tap **Create Account** at the bottom of the screen:

2 On the screen that opens, type your name and email address, type a password for your account, then tap **Create Account**. You will be asked to accept the Terms of Service before continuing.

3 Remove your Modulo development board from its packaging. Using a standard Micro-USB cable, connect the Modulo to any available USB port to supply power to the board.

4 In the mobile app, you will be prompted to add a device to your account. Tap **Found it? Tap to scan.** If you are prompted to allow the app access to the Camera, please allow access. On the screen that appears, tap **Connect New Device** at the bottom of the screen:

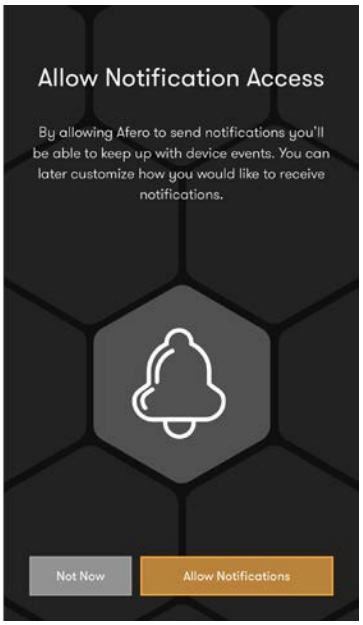




5 Use the Camera view in the app to scan the QR code sticker on the back of the Modulo board.

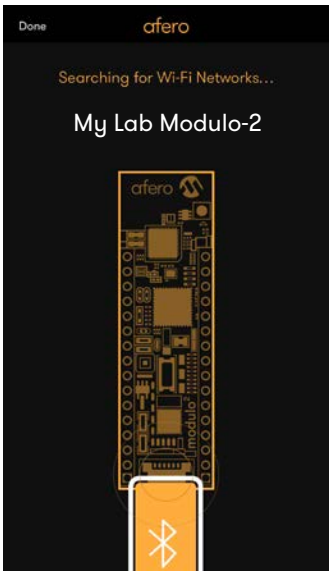
TIPS
- ✓ If you're not prompted to add a device after creating an account, but are brought to the application's main menu, tap the icon at the top-left of the screen to open the app Settings screen, then tap **Add Device** in the menu.
- ✓ If you are unable to scan the QR code with your phone's camera, tap **Manually Add Device** at the bottom of the screen. Type the hexadecimal code printed under the QR code (don't type the dashes!), then tap **Add**.

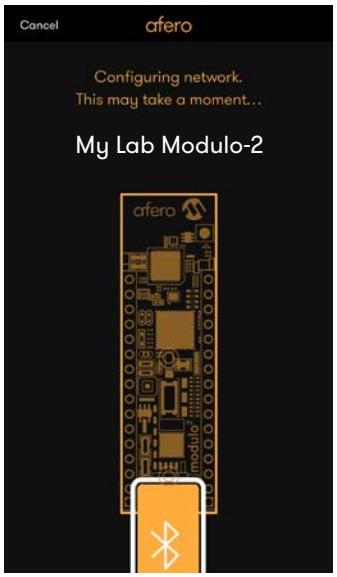
6 You will be prompted to allow Location and Notification permissions on your mobile device. Please allow these permissions to allow the app to provide all its features:



7 Next, give your Modulo board a friendly name so you can identify it if you have multiple devices.

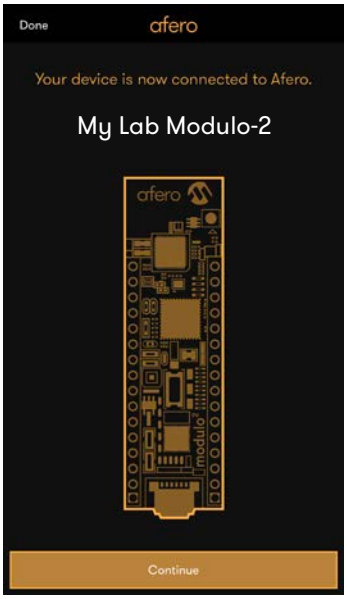
8 The Afero mobile app will instruct the Modulo-2 board to scan for available Wi-Fi networks. You will be prompted to select a Wi-Fi network, then type a password to connect to it. Once the mobile app has connected to a network, the Modulo-2 board will attempt to connect directly to the Afero Cloud to ensure it can communicate over the Wi-Fi network.

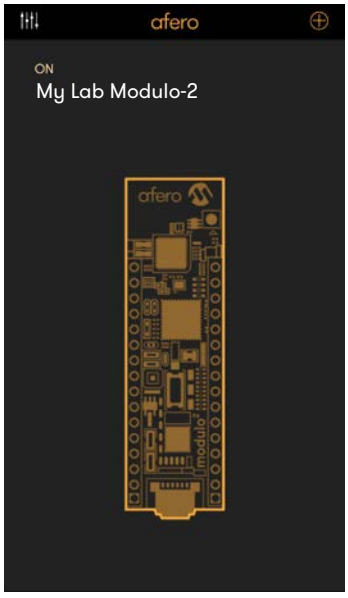




TIP ✓ If any part of this process fails, you will be given the chance to rescan and reconnect to the Wi-Fi network. If no Wi-Fi network is available, you can click **Cancel** and then the board will connect only over Bluetooth® low energy. In BLE mode, the board will require an Afero compatible hub device or an active Afero mobile app to connect to the Afero Cloud, so make sure the mobile app is running (and that your device isn't asleep) when interacting with the Modulo-2.

9 Once your device connects, tap **Continue** and you will return to application's main screen:

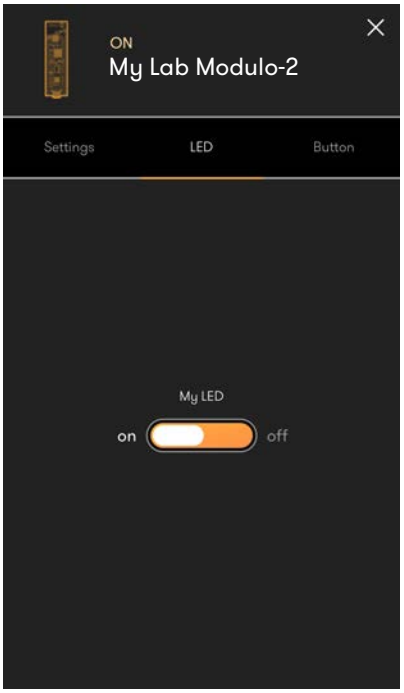


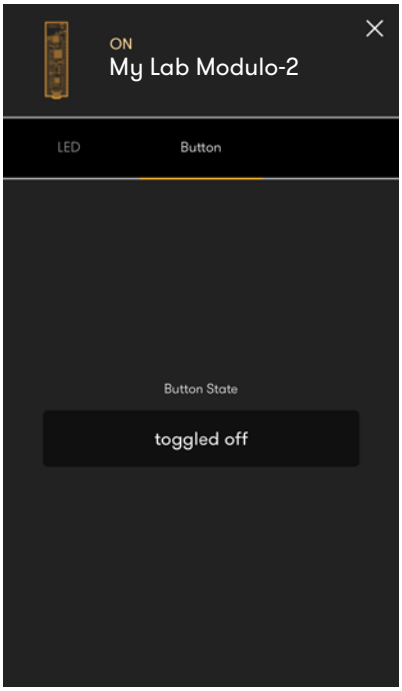


Note that the Modulo icon is an orange color, indicating the device is online and connected to the Afero Cloud. (The icon will be grey if the device is not connected.)

10 Tap the **Modulo icon** to bring up the device screen so you can interact with the Modulo.

Notice the “groups ribbon” running horizontally at the top of the screen; swipe left/right to reveal the groups you created, as well as to access the Settings and Automation screens. Below the groups ribbon, the selected group’s controls are displayed.





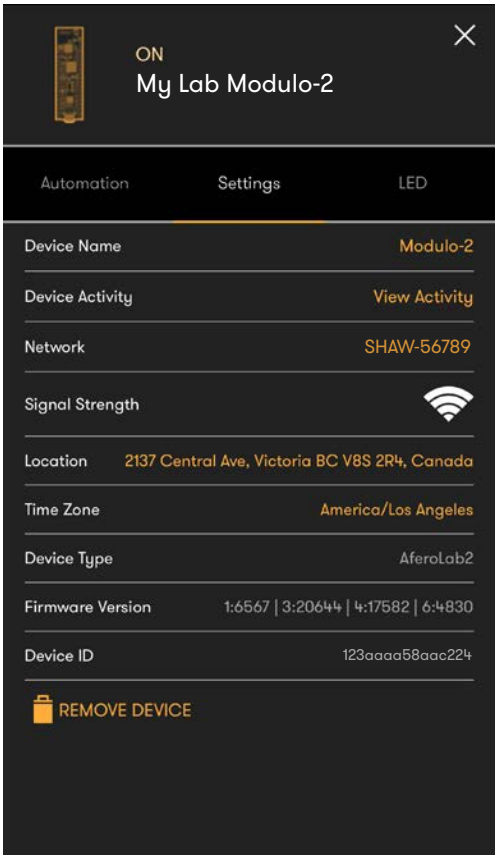
11 Select the LED group from the groups ribbon, then tap **On** and **Off** and observe that the Modulo LED turns on and off. Also note that the device icon will change color when the LED is on. This happens because the LED on/off control has been defined as the device’s “primary operation”, which we’ll talk about more in thein the Using the Afero Profile Editor lab.

Select the **Button** group from the groups ribbon, then press the button on the Modulo. Notice how the mobile app UI updates the status of the button as you press it.

12 Tap **Settings**, located in the groups ribbon. On this screen you can:

- Change the friendly name of the device.
- Change the Wi-Fi network the device connects to.
- Edit the device’s location.

You can also tap **Remove Device** to disassociate the device from your account. If you do remove the device, to use the device again, you will need to repeat the steps above to add the device back to your account.



TIP ✓ The Modulo development board does not have a GPS or any way to determine its location on its own. The location set here is completely controllable by you, and can be used to identify similar devices that may be geographically separated. The location will not change if the device is physically relocated unless you change this location later in the device's Settings.

13 Tap **Automation**, also located in the groups ribbon. Using the Automation screens you can create simple automation tasks for your Modulo, allowing you to run device tasks without having the mobile app running.

1.5 Behind the Scenes

The time between turning on the LED via the mobile app and the device's response seems fast enough to think that the mobile phone is communicating directly with the Modulo board, but this is not the case!

The command to turn the LED on travels from the mobile device to the Afero Cloud (via Wi-Fi or LTE on the mobile device), and then the Afero Cloud sends the command back to the Modulo-2 over its Wi-Fi connection.

For the Modulo-2, as long as the board is connected to the network, these interactions will work no matter how far away the devices are.

1.6 Extra Credit for Modulo-2 Boards

If you have time after completing the basics of this lab, here are some ways you can explore additional features of the Modulo-2 and the Afero Cloud:

1. Turn off Wi-Fi on your mobile device so that it and the Modulo-2 are connected to separate networks (Modulo-2 on Wi-Fi, mobile on LTE). Verify that the device interactions work regardless.
2. Leave your Modulo-2 and take your mobile device to a different location. Have someone near the Modulo-2 press the button and watch while the mobile app updates the button state even though it's physically located elsewhere.
3. Tap the **Automation** menu and experiment with simple automation steps. Try things like:
   - Have your device turn on its LED at a specific time and notify you that it's done so. (**Hint:** You will have to exit the mobile app to receive notifications.)
   - Add a trigger automation to turn the LED on when the button is toggled, and another trigger automation to turn the LED off when the button is toggled off.
   - If you have more than one Modulo-2 board handy, add them to the same account and create a trigger automation so the button on one Modulo-2 will toggle the LED on another. (**Hint:** You'll need two automation events, button on = LED on, and button off = LED off.)

Lab 2. Using the Afero Profile Editor

This lab will demonstrate:

- How to create a simple profile for a Modulo development board.
- How to configure specific behaviors from the available GPIO pins.
- How to create and modify UI elements that present the device in the Afero mobile app.

This lab should take about an hour to complete, depending on how much experimentation you want to do.

2.1 Prerequisites

Before you begin this lab, you should have completed Lab 1. Afero Onboarding, so you have:

- An account on the Afero Cloud.
- A Modulo development board added to your account, connected and online.

2.2 Tools/Materials

For this lab, you will need:

- Windows or Macintosh computer
- iOS or Android mobile phone, running:
  - Android KitKat 4.4 (API Level 19), or
  - iOS release 9.3
- Afero Modulo development board
- Micro-USB cable to power the Modulo
- USB power source (free USB port on a computer, AC wall adapter, or portable charger)

2.3 Setup

1 Download the Afero Profile Editor application to your computer. You can download the Profile Editor installer from the following locations:

   - Windows: https://cdn.afero.io/latest-ape/win
   - macOS: https://cdn.afero.io/latest-ape/mac

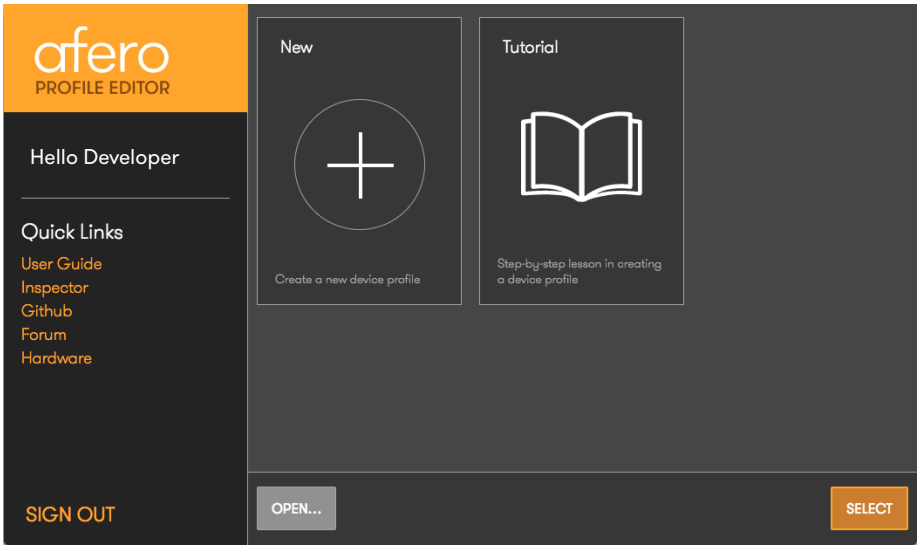
2 Double-click the installer to run it. Installation is automatic and won't require anything from you other than granting permission for the installer to run.

2.4 Steps

1 Launch the Afero Profile Editor on your computer from the Start menu in Windows or from the Applications folder in macOS.

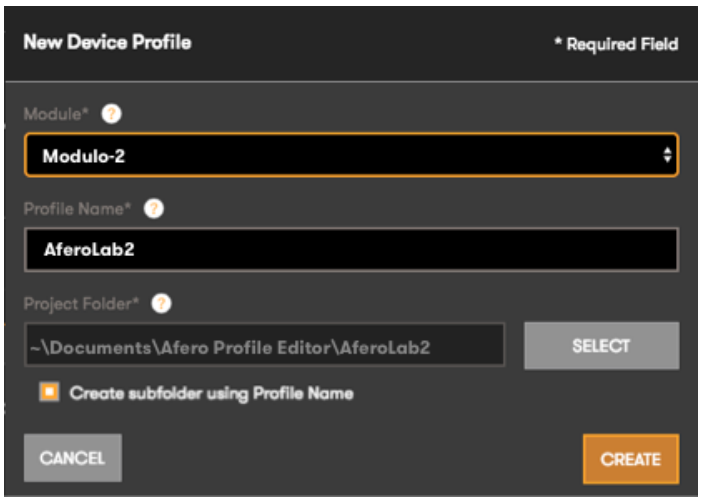
2 On the sign-in window, sign in to the Afero Cloud with the *same* credentials you used in the Afero mobile app. This will allow you to send profiles from the Profile Editor to the devices connected to your Afero account.

3 Click the **+** icon to create a new profile:



You will need to provide:

- The type of Modulo board you're creating a profile. Select **Modulo-1 or Modulo-2 based on the board you have**. (Hint: Modulo-1 boards have a square silver chip on the board with an Afero logo with the letters "af". Modulo-2 boards will not have this silver modulo and will be labeled "Modulo^2" printed on the board.)
- The name of the profile (this will also be called the "device type" since we're defining what the device is going to be: toaster, mousetrap, etc.). Use any name you like, or something descriptive like **AferoLab2**.
- The location of the folder where you want to save the profile (by default this should be in your **Documents** folder, or you can click **Select** to put the profile in a folder of your choosing.) By default, the Profile Editor will create a folder using the Profile Name to store the files created.
- Click "Create" to create your new profile and start customizing it.

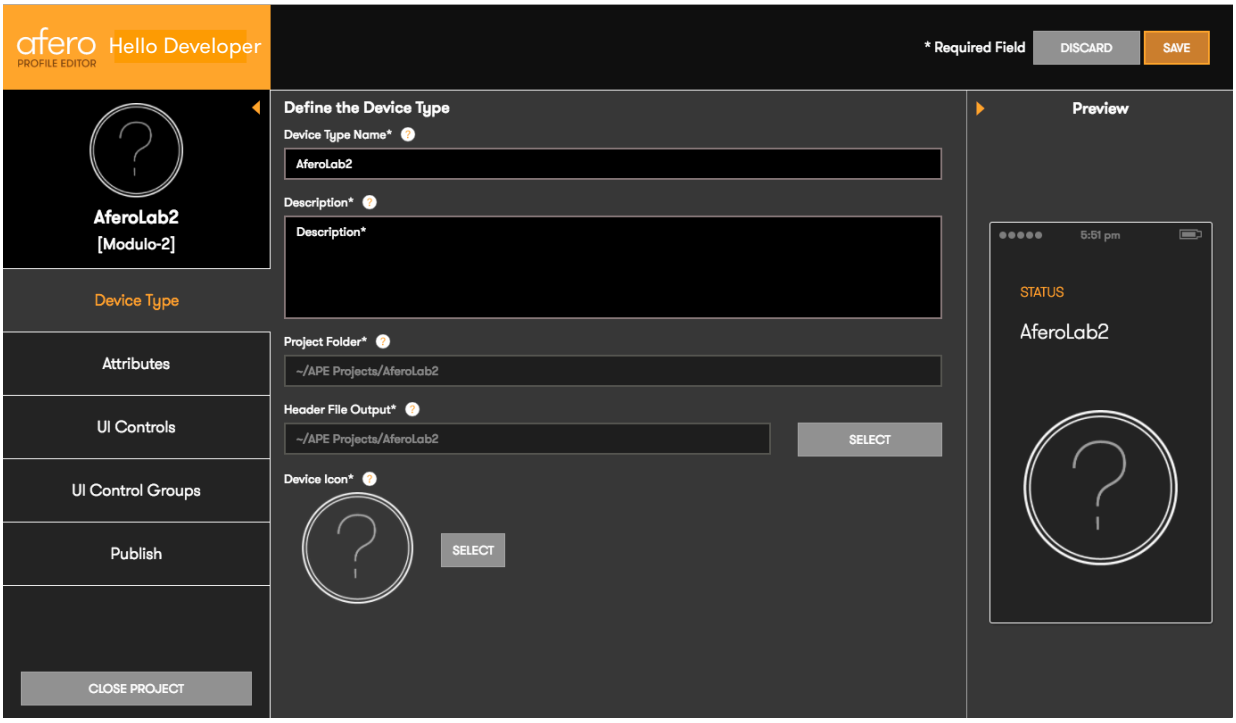


4 A navigation bar, containing a list of options, runs vertically along the left side of the window. We'll run through all of these items to produce a profile.

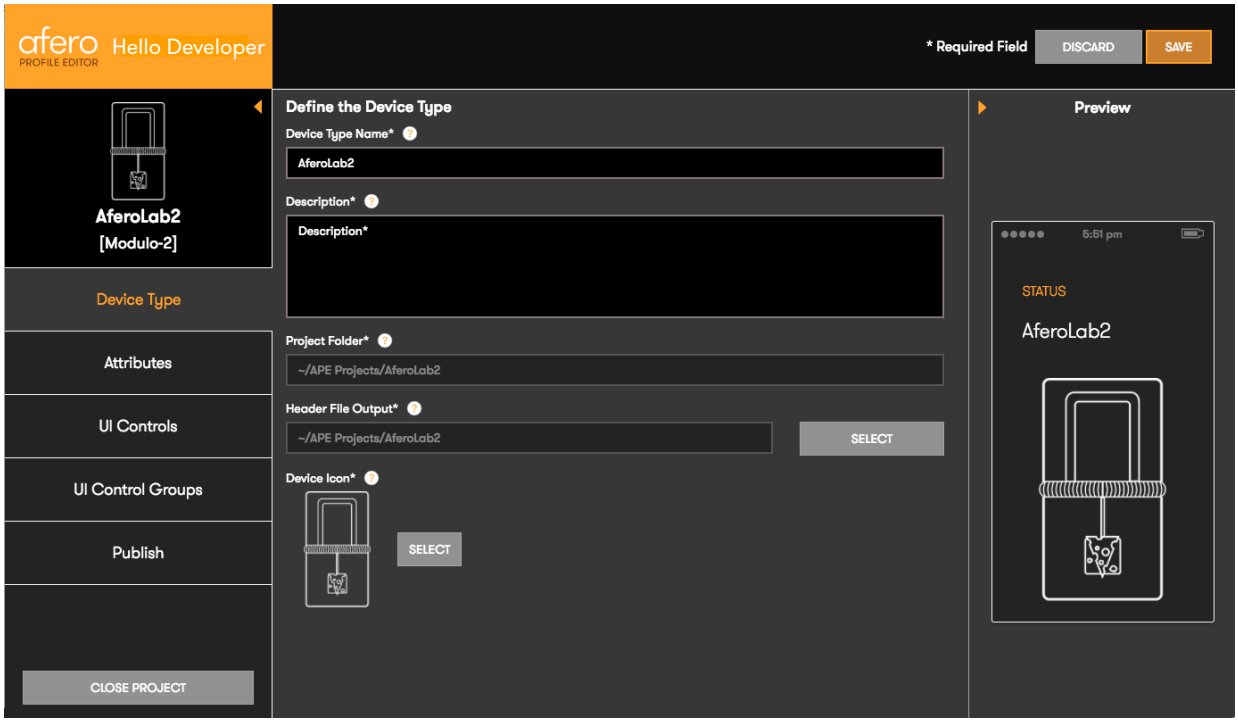
- Device Type
- Attributes
- UI Controls
- UI Control Groups
- Publish

5 **Device Type Window**

a. In the Navigation pane at the left, **Device Type** will be selected by default. On this window much of the information will be populated from the New Profile dialog you just completed.



b. Click the **Select** button to the right of **Device Icon**. The icon you select will be the icon displayed for your device on the Afero mobile app. Select an icon from the provided list. In the lab examples, we've selected a mousetrap.



c. To save your settings and advance, click **Save** in the upper-right corner of the Profile Editor window.

6 Attributes Window

In the Navigation pane at the left, click **Attributes**. The Attributes window is where we define the pieces of data that are synced between the device, the Afero Cloud, and the Afero mobile app.

> **NOTE** We'll use the term "attribute" a lot from here on. The easiest way to think about an attribute is that it's the smallest *useful* piece of data that can be communicated between the device and the cloud.
>
> **Good attribute examples:** Current Temperature, Target Temperature, Switch State, Altitude, Direction, Sensor Value, Light Level, Error Code, etc.
>
> Attributes should not be constantly-changing variables, since there is overhead in sending that data to the Afero Cloud. Avoid "live" updates, such as: GPS Position, Accelerometer, Velocity, RPM. Instead send point-in-time snapshots of such data.

a. At the top of the Attributes window you can select an MCU protocol; we support MCU attachment via SPI or UART, which we'll get to in a later lab. For now, leave that as **No MCU** and we'll configure a couple of GPIO pins to tinker with.

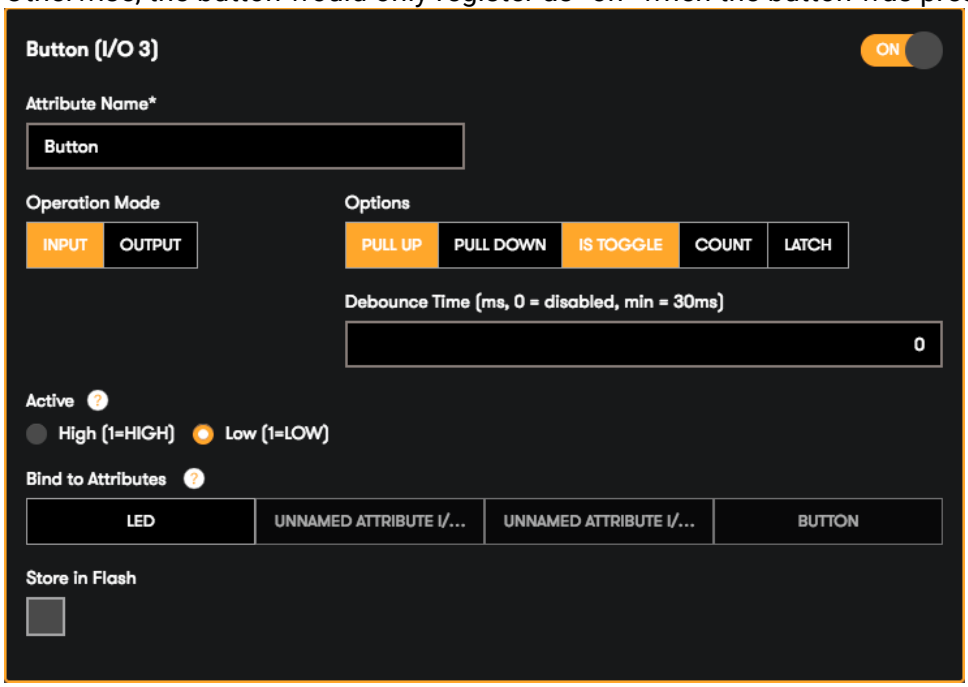
b. Turn on the **Unnamed Attribute I/O 0** switch. GPIO 0 on the Modulo board is connected to the on-board LED, so let's define this GPIO as an output:

- Attribute Name: **LED**
- Operation Mode: **Output**
- Default Logic Level: **0**
- Operation Mode: **Output** (leave all the Options deselected for now)
- Active: **Low (1=Low)**

c. Turn on the switch for **Unnamed Attribute I/O 3**. GPIO 3 on the Modulo board is connected to the on-board button, so let's define this GPIO as an input:

- Attribute Name: **Button**
- Operation Mode: **Input**
- Options: **Pull Up** and **Is Toggle**
- Debounce Time: **0**
- Active: **Low (1=LOW)**

Setting the button as **Toggle** will let us turn that GPIO on and off with each button press. Otherwise, the button would only register as "on" when the button was pressed and held.



d. Click **Save** in the upper-right corner of the window.

7 UI Controls Window

Now that we've defined the attributes in our profile, we can create UI controls that will display the state of those attributes in the mobile app.

a. In the Navigation pane at the left, click **UI Controls,** then click **+New Control** at the top of the window.

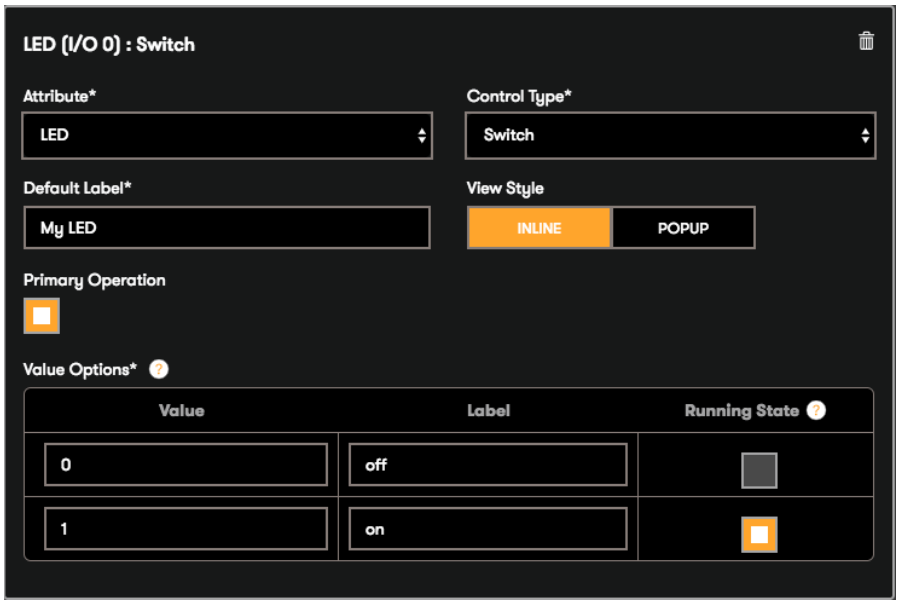
b. We provide a toolbox of simple UI elements you can use to display attributes in the mobile app. For the LED, select **Switch**, then click **Add**.

- Select **LED** under the Attribute drop-down.
- The **Default Label** is a text description that is shown above the UI Control in the app to describe what the control does. Call this one **My LED** or whatever you like, so you can identify where it shows up in the mobile app.
- For **View Style**, select **Inline**. Inline style exposes each menu item as a selectable button. Popup style shows the current value in a selectable circle, which expands to the full menu on tap. Popup is good if you have a lot of controls because it's more compact.
- Select the **Primary Operation** checkbox. This checkbox is used in the mobile UI to indicate that the device is performing its defined task. When the profile's "primary operation" is running, the device's icon will turn orange so you can easily tell the device is doing something.
- The **Value Options** section allows you to map attribute values to meaningful descriptions of the values. For example, the GPIO pin attached to our LED only has two states: **1** (on) and **0** (off). In general, if we don't put anything in the Value Options for a control, the "raw" attribute value will be displayed, which may not be intuitive to the mobile user.

  Running State is used with **Primary Operation** to indicate when the device is actively performing its main task.

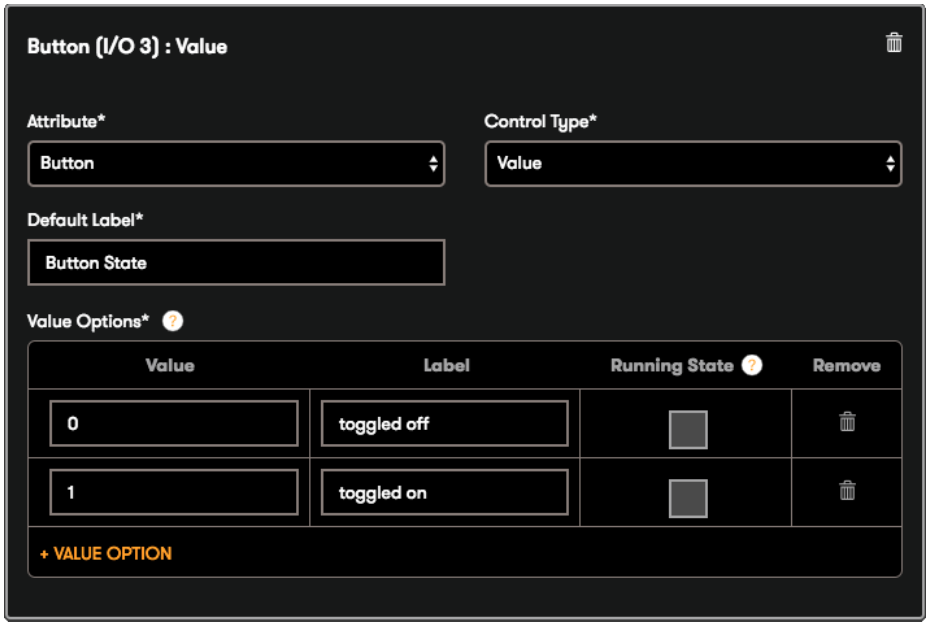
  For a Switch UI element we must provide two Value Options that provide human-readable labels describing each state, defined as follows:

  - 0=**off**
  - 1= **on**, with Running State selected



c. Click **+New Control** to create a new UI control for the button:

- Select **Value** as the UI element type, which is a simple text box we can use to display the button state.
- Define this UI Control as follows:
  - Attribute: **Button**
  - Default Label: **Button State**
  - Value Options, define two: 0 = **toggled off**; 1 = **toggled on**



d. Click **Save** in the upper-right corner.

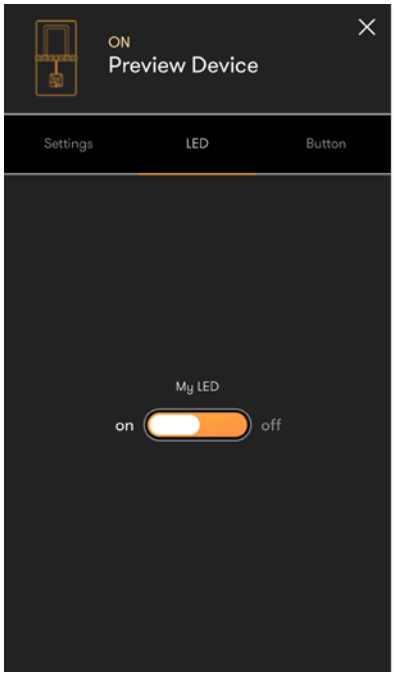
8 **UI Control Groups Window**

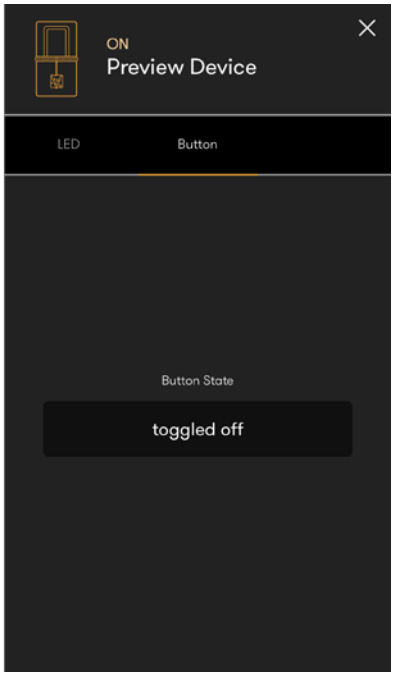
Groups allow us to organize UI controls into logical groups. When you interacted with the Modulo earlier, there was a control for LED that contained the LED control buttons (menu), and another control for Button that displayed the button state. The layout – groupings – for those controls are defined here in UI Control Groups.

a. In the Navigation pane, click **UI Control Groups**. A mockup of the Afero mobile app is displayed. Along the top there is the "groups ribbon". It's empty to start so we'll click to create the first group. Click the "New Group" name and type "LED". This will be the group that holds the LED control.

b. From the right, under **Available UI Controls**, drag & drop the LED switch to the area under the groups ribbon. Create another group named "Button" and drag the Button State control into the area under the groups ribbon.

c. Click the **Preview UI** button to see how the UI looks on your smartphone. Select the Preview Device from the mobile app Swipe left in the groups ribbon to switch from Settings, LED, and Button controls:





d. Back in the Profile Editor's UI Controls window, click **Save** in the upper-right of the window.

9 **Publish Window**

The Publish menu is where you can deliver this profile over-the-air to any of the devices connected to your account.

a. In the Navigation pane, click **Publish**. Your Modulo will appear in the device list after a brief pause:

b. Ensure the device is connected to power and online. The device **Status** will be orange with a signal strength measurement if the device is online, as shown in the example; or will report "Offline" if the device is not online.

c. Select the checkbox to the left of the Modulo you want to publish this profile to.

d. Click **Publish**! If you're watching the mobile app when this happens, you'll see the device get a short over-the-air software update and then reboot. When the device reboots, the icon and menus will change to the icons, labels, and UI elements you put in your profile. Verify that the LED switch works properly and that the text box for the button changes when you press the button on the Modulo.

2.5 Experiment

Now that you have a basic working profile, if there is remaining time in the lab session, go back and modify your profile in the ways shown below. Publish your profile after each change and observe the difference in the mobile UI and the behavior of the Modulo board:

1 Control LED Blinking

a. Edit the Attribute for GPIO 0 and set the **Pulse** option.

b. Use **500ms active/500ms** inactive.

c. Edit the UI Control for the LED and change it from **Switch** to **Slider**.

d. Set the slider range from 0-10 step 1.

e. Publish your profile.

Instead of turning the LED on and off, now the Modulo will blink the LED (500ms on, 500ms off) as many times as you set the attribute value to. While the LED is blinking, you can change the slider value to **0** and the Modulo will stop blinking.

2 Adjust PWM Cycle

a. Edit the Attribute for GPIO 0 and set the **PWM** option.

b. Leave the **PWM Frequency** at its default.

c. Edit the UI Control for the LED and change the **Slider** range to 0-100 step 10.

d. Publish your profile.

The attribute value (set by the slider) now sets the PWM duty cycle from 0-100%. Adjusting this PWM cycle will alter the brightness of the LED from off (0%) to full on (100%).

3 Use a Menu to Limit Setting Options

a. Edit the UI Control for the LED and change it from **Slider** to **Menu**.

b. Select the **Primary Operation** checkbox.

c. Add the following Value Options to the UI Control (using the **+Value Option** button to add more entries):

VALUE	LABEL	RUNNING STATE
0	Off	Deselected
25	Low	Selected
50	Medium	Selected
75	High	Selected
100	Full	Selected

d. **Publish** your profile.

Notice that you can use multiple Menu options to set specific values for the attribute that you control instead of letting the user set an arbitrary value with the slider.

2.6 Extra Credit

1 Menu vs. Slider

a. Change the GPIO 0 attribute back to **Pulse.**

b. Instead of a slider for the UI Control, use a Menu with several Value Options to allow the user to request 0, 5, 10, 15, or 20 pulses.

2 Count

a. Edit the Attribute for GPIO 3 and change it from **Is Toggle** to **Count**.

b. Edit the UI Control for the **Button** and remove the two **Value Options** from the control.

c. Publish your profile.

Press the button on the Modulo repeatedly and note that the button state keeps a running count of how many times you've pressed the button.

3 Map Value Options to labels

a. Edit the UI Control for the **Button** and add a few **Value Option** entries for the control. Use the examples below, or something similar of your own choosing, to see how attribute values map to human-readable labels:

VALUE	LABEL
0	Button Not Pressed
6	Half Dozen
12	Dozen
13	Baker's Dozen

b. **Publish** your profile.

Press the button every second or so for at least 13 times. Note that as you press the button, the raw value of the attribute is displayed in the UI unless there's a specific Value Option entry for that value. This way you can map specific values to more meaningful terms if you need to.

2.7 Practical Uses

You can see that for simple control or sensing tasks, the Modulo can be used on its own, without an attached MCU. Using just GPIO attributes and various hardware options you could create simple sensors or controllers:

- Use a magnet and a Hall effect sensor and a Count attribute to count the number of times a door has been opened or closed.
- Use two GPIOs with Pulse attributes connected to a motor controller to turn a simple motor or servo in either direction.
- Use an output attribute with Toggle connected to a relay to control a larger electric device such as a lamp or fan.
- Use an output attribute with PWM connected to a small vibration motor (like a pager or cellphone vibration motor) to "buzz" the device at different pitches for different haptic notifications.

- Use an input GPIO with Toggle connected to a piece of copper tape, and another piece of copper tape connected to the Ground pin of the Modulo, and use the two pieces of tape as a water sensor under a sink or water tank.

For more complex tasks, the Modulo can be connected to an MCU and then it can be used to communicate arbitrary data between the MCU and the Afero Cloud. The next lab will expand your new knowledge of the Afero Profile Editor as we connect a microcontroller to the Afero Cloud.

Lab 3. Afero + MCU = Arduino Temperature Sensor

This lab will demonstrate how to:

- Connect a Modulo development board to a simple microcontroller.
- Send data from the MCU to the Afero Cloud.
- Receive data from the Afero Cloud to the MCU.

This lab will take about an hour, depending on your familiarity with the popular Arduino MCU and its development environment. No previous experience with the Arduino MCU is required to complete this lab.

NOTE This lab uses afLib2, written in C. The C++ version of afLib has now been deprecated; however, Lab 4 still is based on this deprecated version.

3.1 Prerequisites

You should have completed Using the Afero Profile Editor, and be able to modify a profile and publish it to a Modulo development board over-the-air.

3.2 Tools/Materials

Before beginning this lab, you will need:

- The following pieces of hardware supplied in your lab hardware kit:
  - Arduino Uno MCU
  - Afero Plinto interface board to connect your Modulo to the Arduino Uno
  - MCP9700 Temperature Sensor (small component with three wire leads)
  - USB-A to USB-B cable to connect the Arduino to your computer
- Windows or Macintosh computer
- iOS or Android mobile phone
  - Android KitKat 4.4 (API Level 19)
  - iOS release 9.3
- Afero Modulo development board
- Micro-USB cable to power the Modulo
- USB power source (free USB port on a computer, AC wall adapter, or portable charger)

3.3 Arduino IDE Setup

1 Download the Arduino IDE from the web:

- Windows: https://www.arduino.cc/download_handler.php?f=/arduino-1.8.3-windows.exe
- macOS: https://www.arduino.cc/download_handler.php?f=/arduino-1.8.3-macosx.zip

2 Install the Arduino IDE following their online instructions:

- Windows: https://www.arduino.cc/en/Guide/Windows
- macOS: https://www.arduino.cc/en/Guide/MacOSX

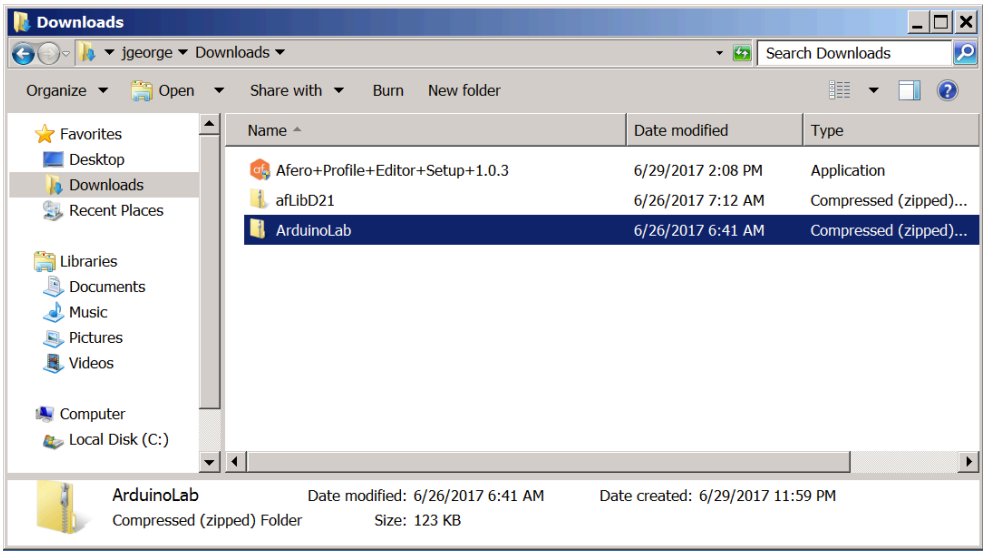
3 You should have received a file download as part of this lab named ArduinoLab.zip. Save this file to your computer's **Downloads** folder. You can also download the lab project file from https://developer.afero.io/static/custom/files/ArduinoLab.zip.

Copy the files in this .zip file to your computer into the specific locations listed below. Please note that it is important to put these folders in the locations specified so the Arduino IDE can properly locate the code needed for the lab.

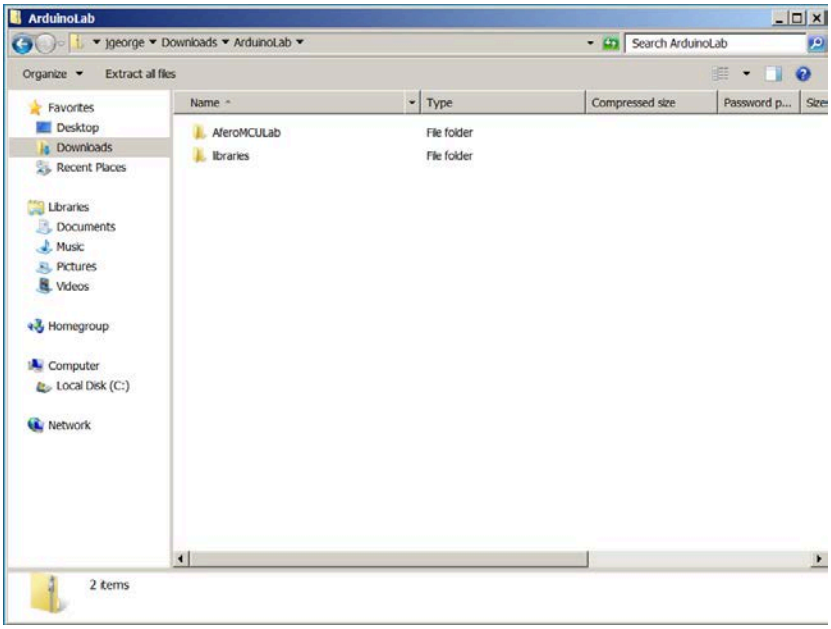
3.3.1 Arduino IDE Setup for Windows

1 Open a File Explorer window either by clicking the File Explorer icon in the taskbar or selecting **Computer** from the Start Menu.

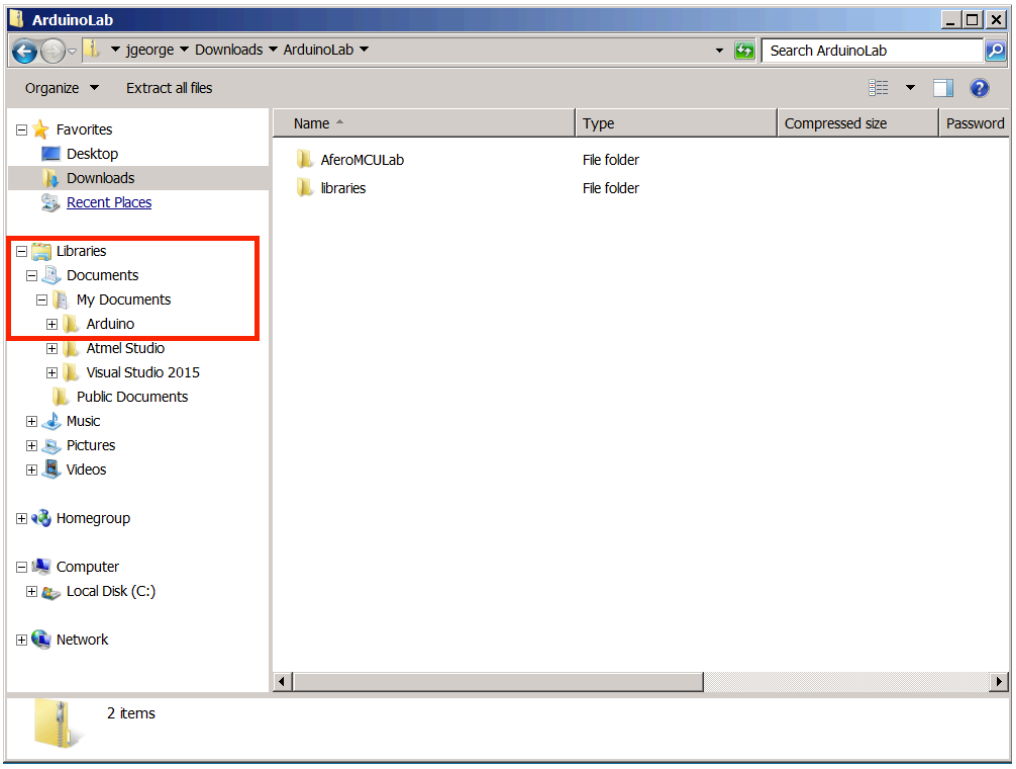
2 Click the **Downloads** folder in the left pane, and find the file named **ArduinoLab**.



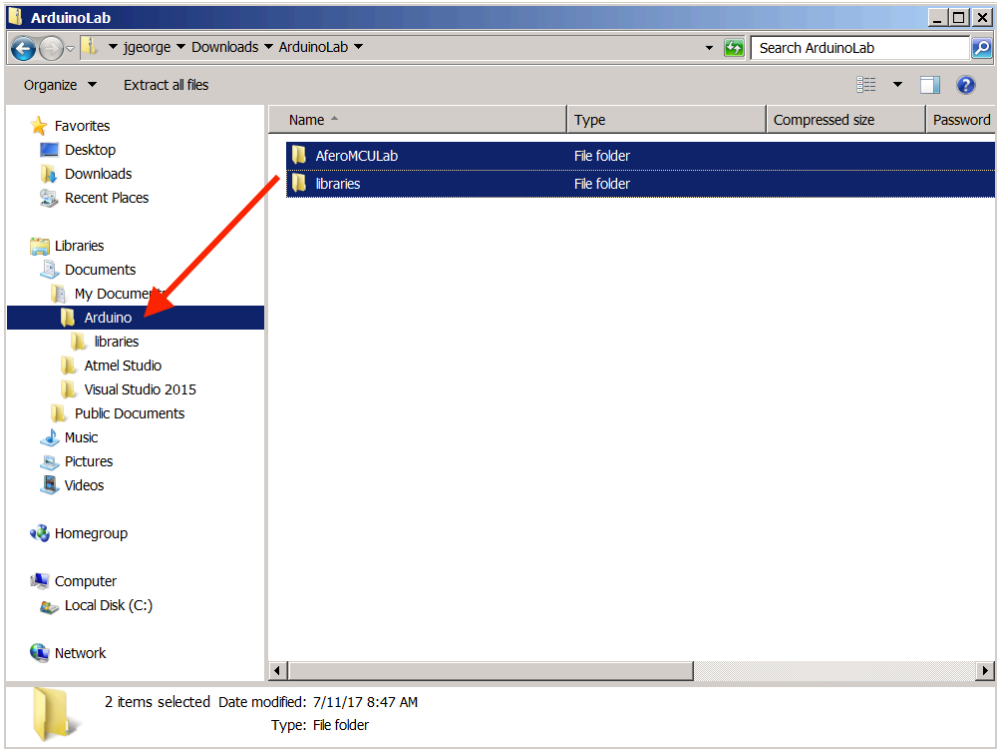
3 Double-click the **ArduinoLab** compressed folder. In that file you will see two folders, one named **libraries** and another named **AferoMCULab**.



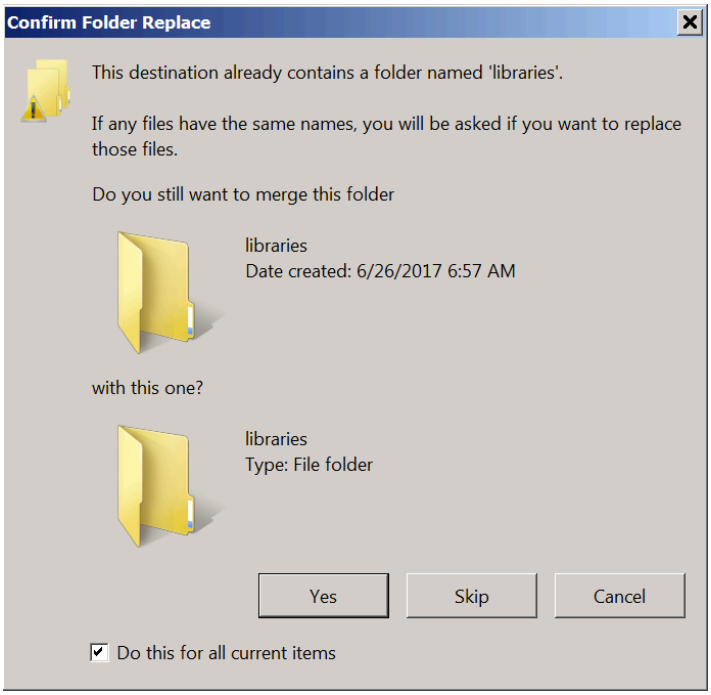
4 In the left pane of the File Explorer window, click the **+** sign next to the **Documents** folder (under **Libraries**) and then click the **+** sign next to the **My Documents** folder displayed. In the list you will see a folder called **Arduino**, which was created when the Arduino IDE was installed earlier.



5 Drag **both** folders in the right pane (**libraries** and **AferoMCULab**) to the Arduino folder shown in the expanded list on the left.



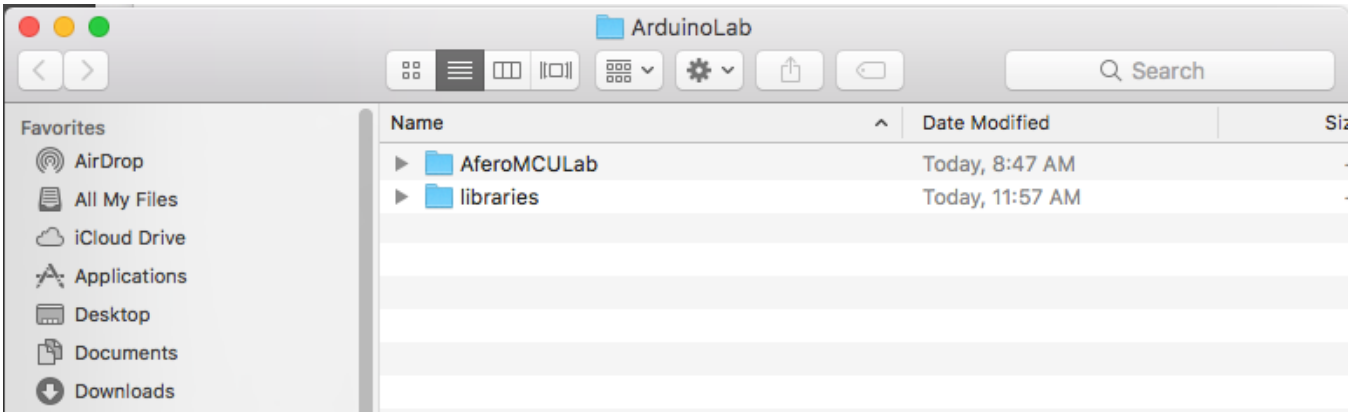
6 You will be warned that there is an existing folder named **libraries** and asked if you want to merge the two. Select the **Do this for all current items** checkbox at the bottom of the window, then click **Yes**.



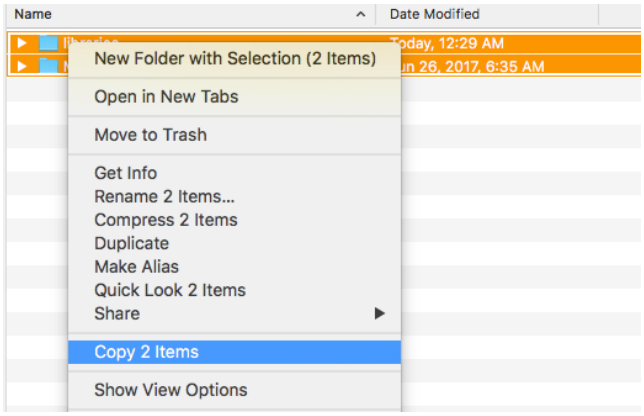
7 You are now ready to continue with the lab. Skip to the next section, Create Device Profile.

3.3.2 Arduino IDE Setup for macOS

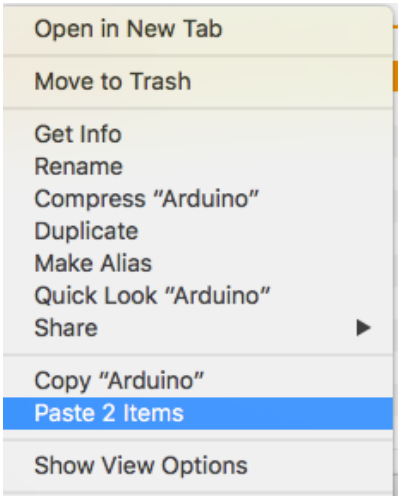
1 Open a Finder window by clicking on the Finder icon on the left side of the dock. Click the **Downloads** folder and then double-click the **ArduinoLab.zip** file you downloaded earlier. A new folder named **ArduinoLab** will be created in your Downloads folder. Double-click that folder to open it, and you will see two folders named **libraries** and **AferoMCULab**.



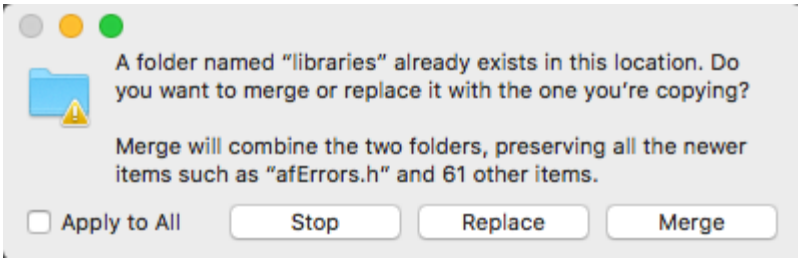
2 Select both folders, right-click, then select **Copy 2 Items**.



3 In the left pane of the Finder window, click the **Documents** folder. In this folder you will see a folder named **Arduino**. This folder was created when the Arduino IDE was installed earlier. Right-click the Arduino folder, and select the menu option **Paste 2 Items**.



4 You will be warned that a folder named **libraries** already exists in this location. Select the **Apply to all** checkbox and then click the **Merge** button.



5 You are now ready to continue with the lab. You can skip to the section, Create Device Profile.

3.4 Steps

3.4.1 Short on Time?

The section below will walk you through modifying the profile you've worked with in the previous lab, to add support for this MCU lab. If you're short on time to complete this lab, and are comfortable using the Profile Editor, we've provided a pre-built profile that works with this lab:

1 Open Profile Editor, and navigate to **Documents-Arduino-AferoMCULab-profiles**.

2 If you have a Modulo-2 development board, single-click the **AferoMCULab-2** folder (use AferoMCULab-1 only if you have a Modulo-1 board!), then click **Select Folder**.

3 You can then publish this profile as-is to your Modulo and skip the rest of this section. Otherwise, continue below.

3.4.2 Create Device Profile

On your computer, launch the Afero Profile Editor and open the profile you created in the previous lab. We are going to modify the profile to support MCU connectivity and define a couple attributes, which the MCU will be able to access.

1 In the Profile Editor left-hand Navigation pane, click **Attributes**.

a. At the top of the Attributes window, change the **MCU Protocol** from **No MCU** to **SPI**. A new attribute window titled **Unnamed MCU Attribute 1** appears. We will use this attribute to display the current temperature.

Define this temperature attribute using:

- Attribute Name: **Temperature**
- Default Value: leave blank
- Max Size: **9**
- Data Type: **UTF8S** (UTF-8 String)
- Writeable: **Deselected**

> **TIP** ✓ The **Writeable** checkbox defines whether an MCU attribute can be changed from the mobile app UI. Because this attribute will contain a temperature value read from a hardware sensor, allowing the mobile app to modify it doesn't make sense.

b. Select **+New MCU Attribute** to create another attribute for our application. We will use this attribute to switch the MCU's output from Celsius to Fahrenheit:

- Attribute Name: **Units**
- Default Value: **false**
- Data Type: **BOOLEAN**
- Writeable: **Selected**

c. Click **Save** in the upper-right corner of the window.

2 In the Navigation pane, click **UI Controls**.

a. Click **+ New Control** to add a control using:

- Control Type: **Value**
- Attribute: **Temperature**
- Default Label: **Temperature Sensor**
- View Style: Pick one!
- Value Options: leave blank

b. Click **+ New Control** to add another, this time using:

- Control Type: **Switch**
- Attribute: **Units**
- Default Label: **Scale**
- View Style: Pick one!
- Primary Operation: **Deselected**
- Value Options, define two:
  - Value: **false** Label: **Celsius** Running State: **Deselected**
  - Value: **true** Label: **Fahrenheit** Running State: **Deselected**

c. Click **Save** in the upper-right corner of the window.

3 In the Navigation pane, click **UI Control Groups**.

a. Create a new group by clickimg im the groups ribbon. Name the new group “MCU Lab”.

b. Drag the **Temperature** and **Scale** UI Controls into the area under the groups ribbon.

c. Click **Save**.

4 In the Navigation pane, click **Publish**.

a. Go ahead and publish your profile to your Modulo. Ensure that the Modulo is plugged into a USB power source and online.

b. When the device reboots and the mobile app for the Modulo includes your new MCU Lab menu, unplug the Modulo from the Micro-USB cable so we can assemble the hardware.

3.4.3 Assemble Arduino Hardware

1 Remove the Arduino Uno from its packaging if you haven't already. Remove the Plinto adapter from its packaging and carefully insert the Plinto pins into the sockets on the edges of the Uno's circuit board. Take care that all pins are inserted properly and none are bent.

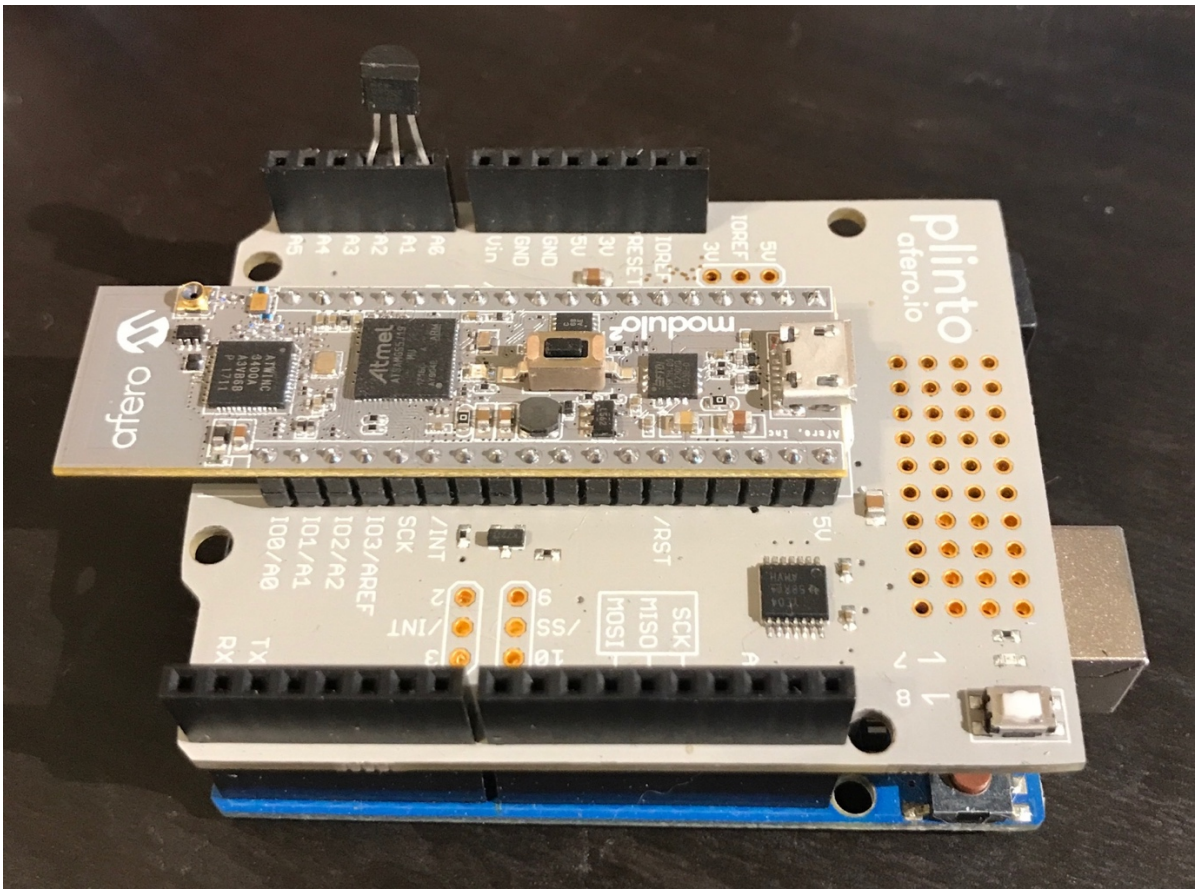
2 Take your Modulo development board and align its pins in the socket on the Plinto. The Modulo USB port should be on the same side as the Arduino Uno's USB port. The end of the Modulo with the Afero and Microchip logos should overhang the edge of the Plinto a little bit. The Modulo will not work properly if it's inserted backwards.

3 Install the MCP9700 Temperature sensor. The sensor has a flat side and a rounded side, with three pins sticking out the bottom of the sensor.

With the flat side of the sensor facing "inward" towards the Modulo, insert the temperature sensor into the Plinto board so that the three pins on the sensor are inserted into the three adjacent pins labeled A0, A1, and A2 on the left side of the Plinto.

You may have to separate the pins on the sensor very slightly to line up the pins with the holes.

IMPORTANT! The temperature sensor will not insert fully into the socket and will stick up around 0.25" above the Plinto socket. This is perfectly normal. Do not force the sensor "all the way" into the pins on the Plinto.



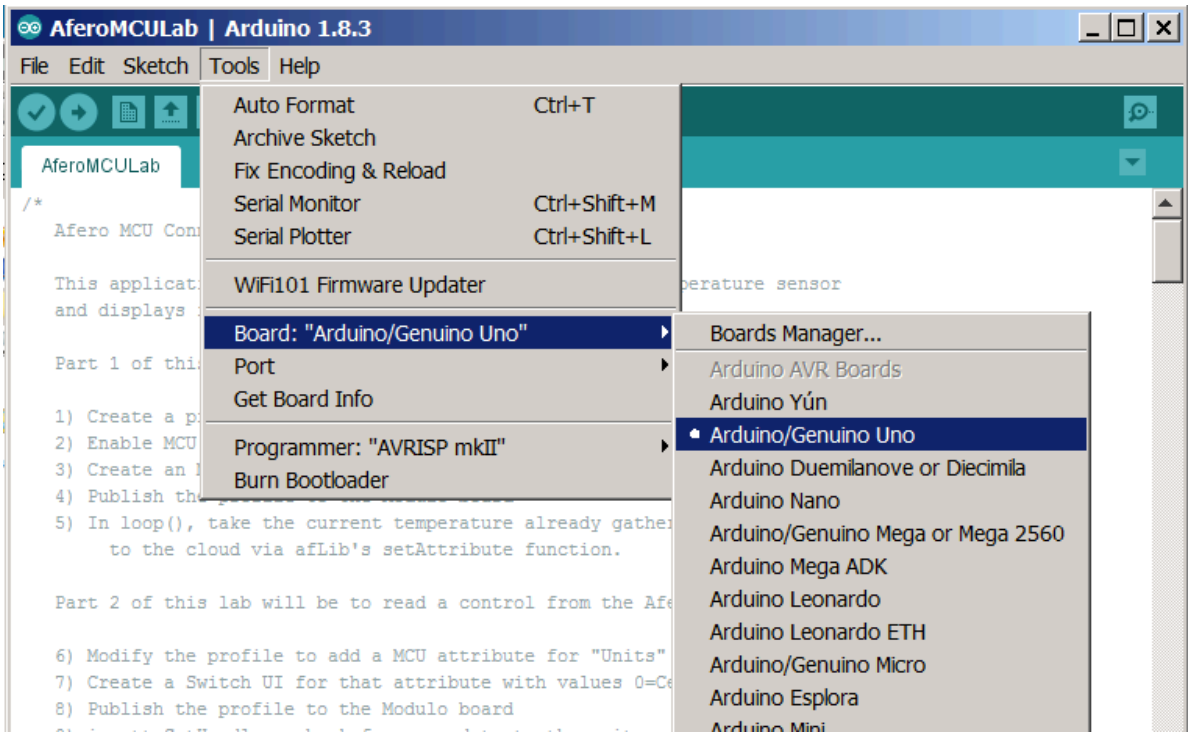
4 Connect the Arduino Uno to your computer with the USB-A to USB-B cable. Windows may display a message that new device driver software is being installed.

3.4.4 Upload Arduino Sketch to Uno

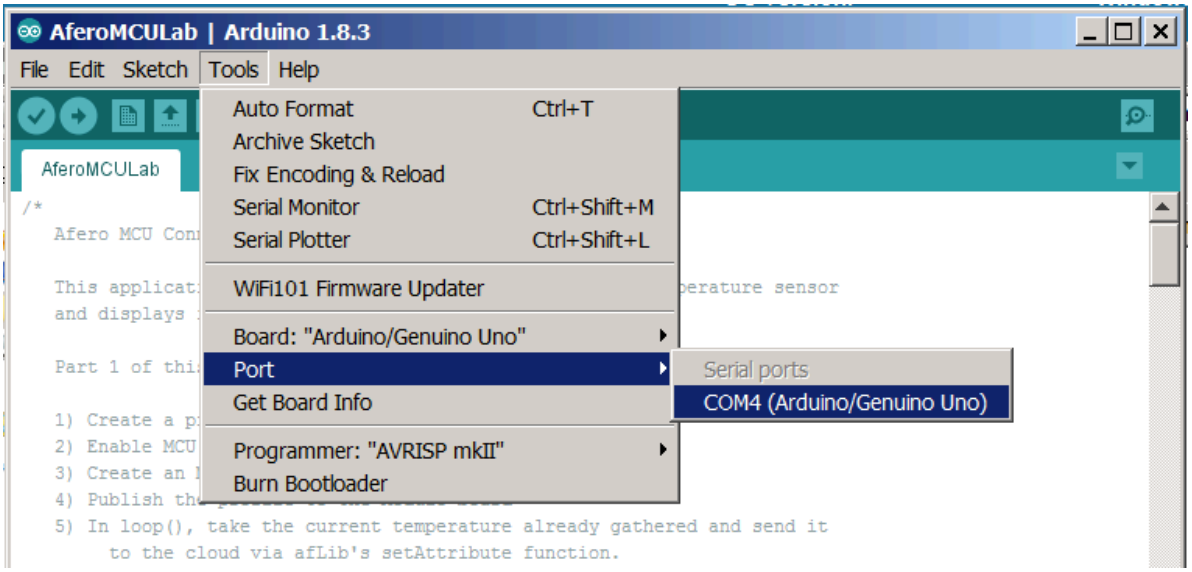
1 Launch the Arduino IDE.

2 From the **File** menu, select **Open** and double-click the **AferoMCULab** folder. Double-click the **AferoMCULab** Arduino file in that folder; the code will be displayed in the IDE.

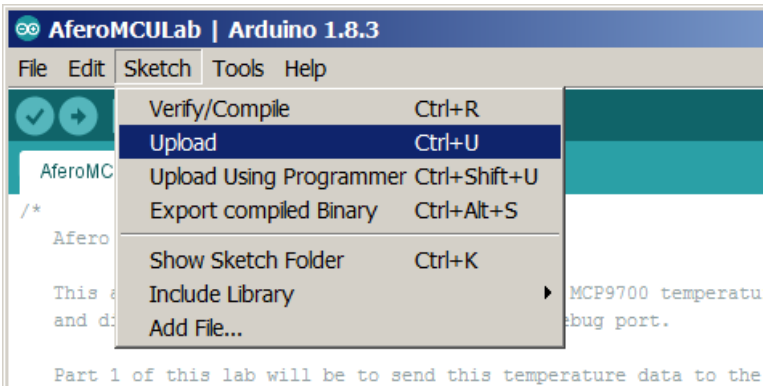
3 In the Tools menu, select **Board,** then select **Arduino/Genuino Uno** as the board type.



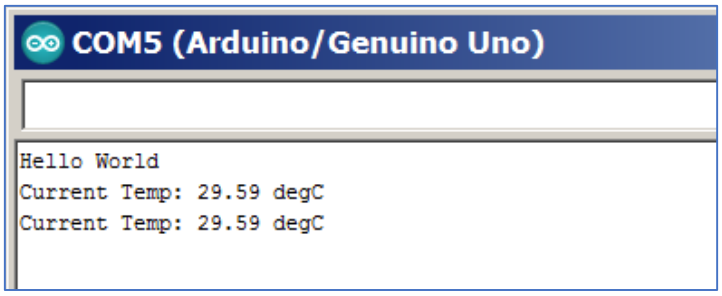
4 In the **Tools** menu, select **Port**, then select **Arduino/Genuino Uno** as the port (the COM port number will be different).



5 Select the **Sketch** menu, and then select **Upload** to compile the code and upload it to the Uno board.



6 When you see the message “Done uploading” at the bottom of the screen, select **Tools** > **Serial Monitor**. Change the baud rate in the lower-right corner of that window to **38400**. You should see messages from the application shortly, including a temperature reading every few seconds.



7 The lab code you just uploaded is a standalone application, like an application you may have that performs a task you might want to connect to the cloud.

8 Now let’s modify the application to send temperature data to the Afero Cloud.

Scroll down in the application code a few screens until you see the **following** section of code in the `loop()` function:

```
/* HINT uncomment this section for lab part one */
//  int rc = 0;
//  String curTempStr;
//  if (fahrenheit) {
//    curTempStr = String(toFahrenheit(currentTemp));
//  } else {
//    curTempStr = String(currentTemp);
//  }
//  rc = af_lib_set_attribute_str(af_lib, AF_TEMPERATURE, curTempStr.length(),
      curTempStr.c_str());
//  if (rc != AF_SUCCESS) {
//    af_logger_print_buffer(F("Couldn't send Temperature update, rc="));
//    af_logger_println_value(rc);
//  }
/* HINT end */
```

Remove the “`//`” (two slashes) from the 10 lines of code between the `/* HINT */` comments.

9 This code handles the selection of Celsius and Fahrenheit, and does a little error checking, but the important part of this code is the line:

```
rc = af_lib_set_attribute_str(af_lib, AF_TEMPERATURE, curTempStr.length(),
    curTempStr.c_str());
```

This call to `af_lib_set_attribute_str` is all that is needed to take the sensor data that we're already reading (`currentTemp` is the value of the hardware temperature sensor) and send it to the Afero Cloud.

10 Scroll down a couple of screens in the code again until you see the following section of code in the `attrSetHandler()` function:

```
/* HINT uncomment this section for lab part two */
//    case AF_UNITS:
//      // if UNITS attribute is 1/true, then display temps in F instead of C
//      fahrenheit = (*value == 1);
//      break;
/* HINT end */
```

Remove the "`//`" (two slashes) from the four lines of code between the `/* HINT */` comments.

`attrSetHandler` is called when the attached Modulo receives an update to an MCU attribute from the Afero Cloud. This function is called to tell your application that some action should occur based on the updated attribute value. In this case, we compare the value of the attribute to "1" (or "true", since it's a Boolean value); and if it's "true", we instruct the Arduino code to convert the temperature sensor value into degrees Fahrenheit from Celsius.

11 Once you've made these changes to the code, select **Upload** from the Sketch menu again, and after the code is compiled and uploaded, look at the device on the mobile app. You should see the temperature being updated every few seconds on the mobile app, and if you change the switch from Celsius to Fahrenheit, the next sensor reading a few seconds later will switch to the updated unit.

3.4.5 Extra Credit

At the very bottom of the Arduino code is a debug method called `printAttribute()`, which can be used to print out attribute values as they're received from the Modulo.

Create a new MCU attribute or two in the Profile Editor, assign UI controls to them, and then add code in `printAttribute()`to print out their values as they're received from the Cloud.

You don't have to add any code to the Arduino sketch if you don't want to, but a little debug code here will demonstrate the variety of data types you can communicate between the Afero Cloud and your MCU.

Lab 4. Atmel XPlained Pro Demo

This lab will demonstrate how to connect a Modulo development board to an Atmel SAMD21 Xplained Pro development board using the Atmel "Afero-Mod2 Xplained Pro" interface board. You will use Atmel Studio 7 to compile a sample application and download it to the SAMD21 Xplained Pro board.

This lab will take about an hour, depending on your familiarity with the SAMD21 Xplained Pro board and the Atmel Studio development environment.

NOTE This lab uses the deprecated version of afLib, written in C++.

4.1 Prerequisites

You should have completed Using the Afero Profile Editor, and be able to modify a profile and publish it to a Modulo development board over-the-air.

4.2 Tools/Materials

For this lab, you will need the following pieces of hardware supplied in your lab hardware kit:

- Atmel SAMD21 Xplained Pro Evaluation Kit
- Atmel "Afero-Mod2" Xplained Pro Interface Board to connect your Modulo to the SAMD21 board
- A computer running Microsoft Windows (go to http://www.atmel.com/tools/atmelstudio.aspx for minimum for Atmel Studio OS requirements)
- iOS or Android mobile phone, running:
  - Android KitKat or later, or
  - iOS 9.3 or later
- Afero Modulo development board
- Micro-USB cable to power the Modulo (there is one in the Afero-Mod2 board kit)
- USB power source (free USB port on a computer, AC wall adapter, or portable charger)

4.3 Atmel Studio Setup

NOTE If you've previously used Atmel Studio on your computer, you can skip to section 4.4 to install the lab files to continue.

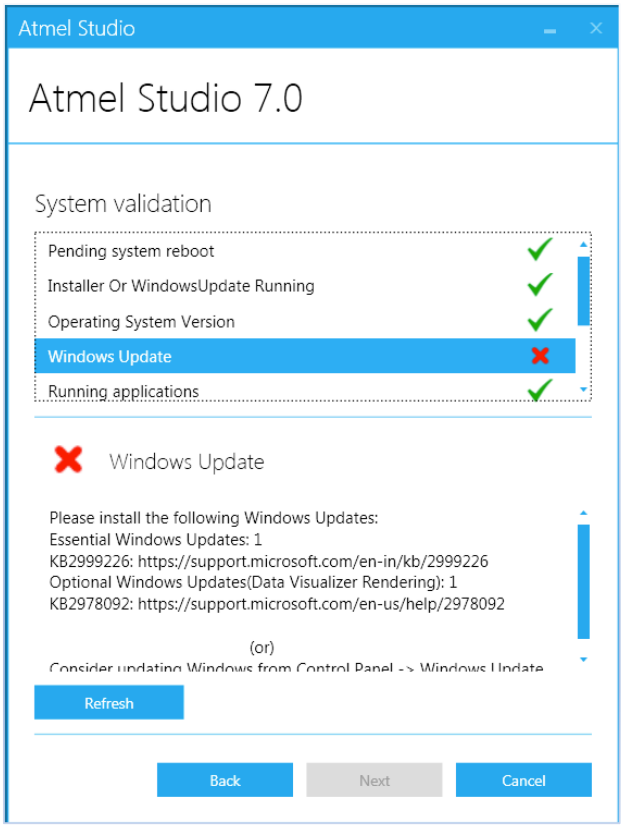
1 Download Atmel Studio 7 from http://www.microchip.com/development-tools/atmel-studio-7.

2 Double-click the Atmel Studio Installer and follow the default prompts to install AS7. You will need to select the **I agree to the license terms and conditions** box on the first page to continue the install. Beyond that leave all the installation prompts at their defaults.

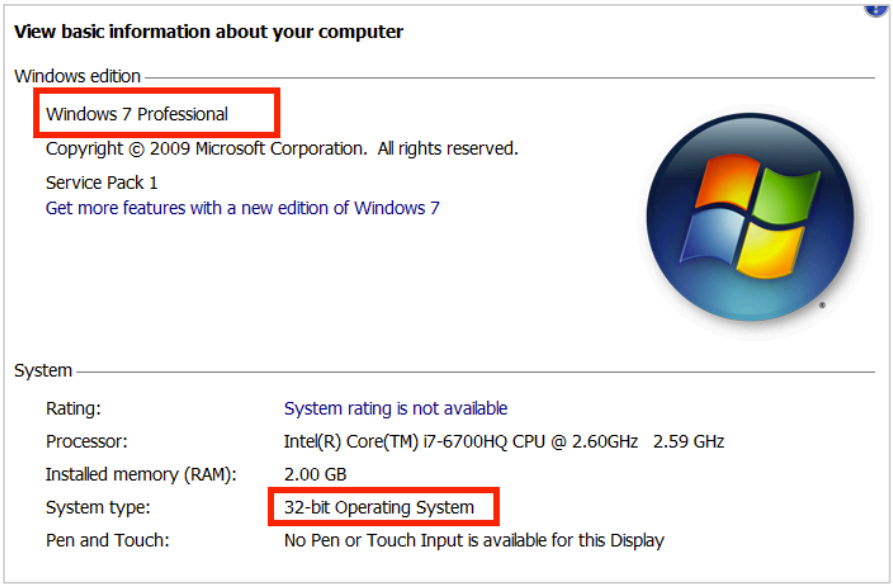
3 Windows 7 users: the installation may pause and request two specific Windows Updates to be installed. You will see the Windows Update installation step fail. If this happens:

   a. The installer will require you to install two updates before Atmel Studio can be installed:

      - KB2999226: https://support.microsoft.com/en-in/kb/2999226
      - KB2978092: https://support.microsoft.com/en-us/help/2978092



b. Copy the two URLs listed in the error message, and paste them into a browser to install the required updates. Select the updates for your specific version and architecture of Windows. Use the system control panel to verify your version of Windows and if it's 32-bit or 64-bit:



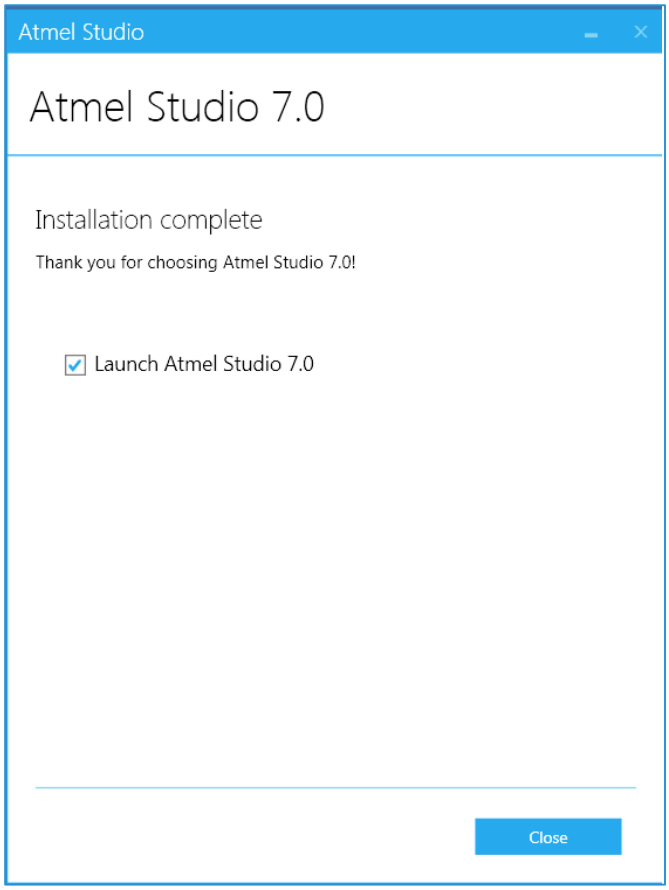
c. Install the two required updates. A reboot is not needed after installation.

d. Click **Refresh** in the Atmel Studio installer to re-run the installation checks, and this time they will pass, so click **Next** to continue the installation.

> **NOTE** If you can't install these updates for any reason, you can click **Next** in the Atmel Studio installer to continue without them. For the purposes of this lab, those updates aren't important.

4 If you receive a note about "Header File Versions" being updated, go ahead and click **Install** on that screen to continue.

5 Continue following the prompts to install Atmel Studio 7 until the installation completes. On the last screen, ensure the **Launch Atmel Studio 7** checkbox is selected and then click **Close** to close the installer and launch AS7.



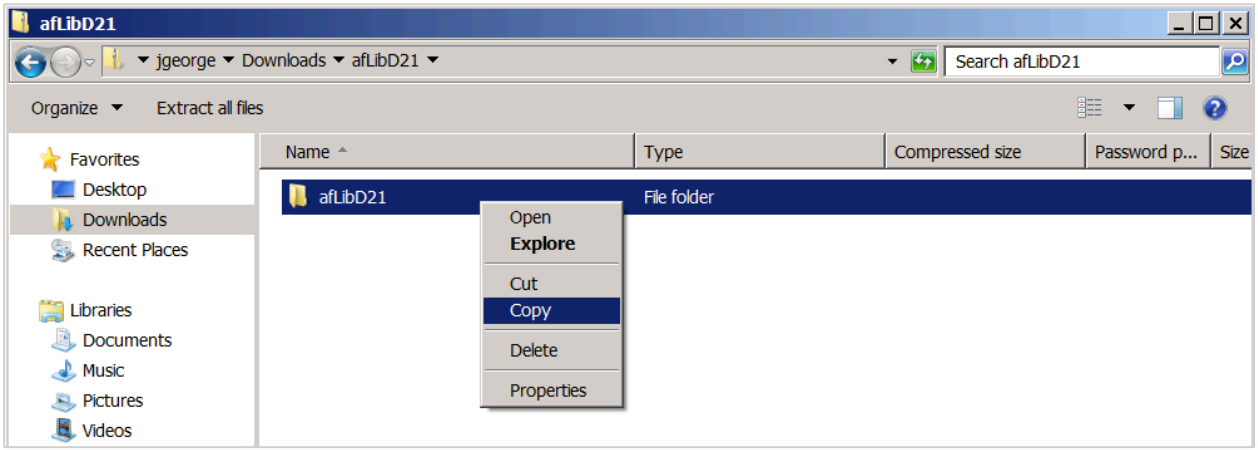
6 When AS7 launches, click the **Minimize** icon in the upper-right corner. Now we can download and install the lab project file and set it up.

4.4 Lab Project Setup

1 Download the lab project file, named **afLibD21.zip**, from https://developer.afero.io/static/custom/files/afLibD21.zip. Save this file to the Downloads folder on your computer.

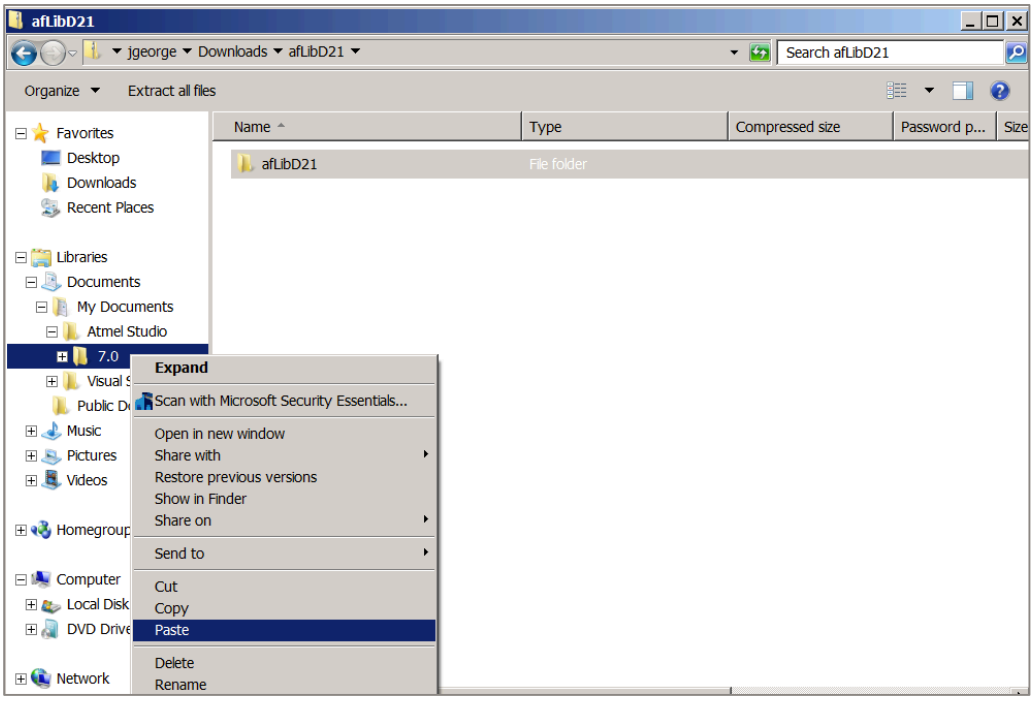
2 Open File Explorer on your computer from the Start menu or from the taskbar. Click the **Downloads** folder in the left pane, then double-click the **afLibD21** compressed folder shown.

3 Inside that file, right-click the **afLibD21** folder and select **Copy**:

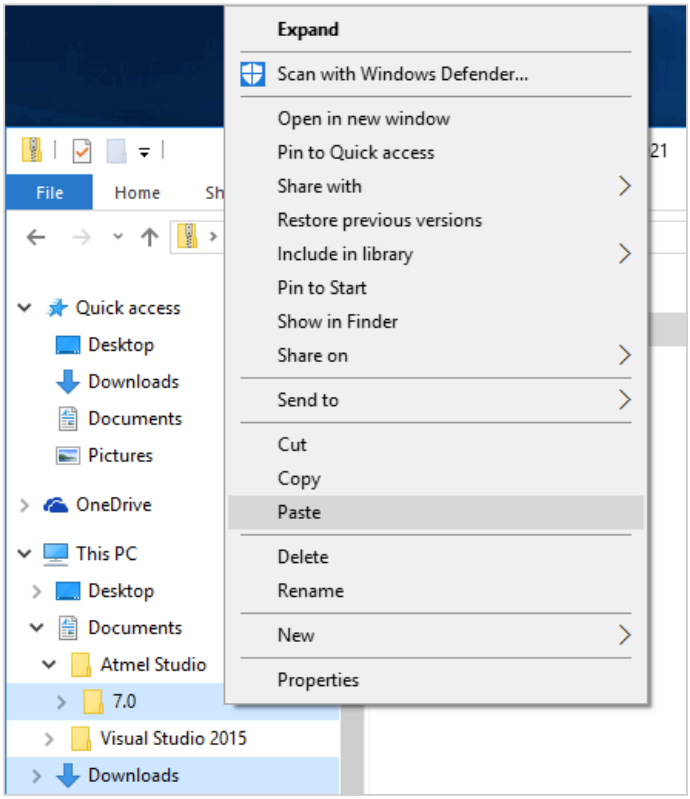


4 In the left pane, expand the folder list until you can see the **Atmel Studio 7** projects folder.

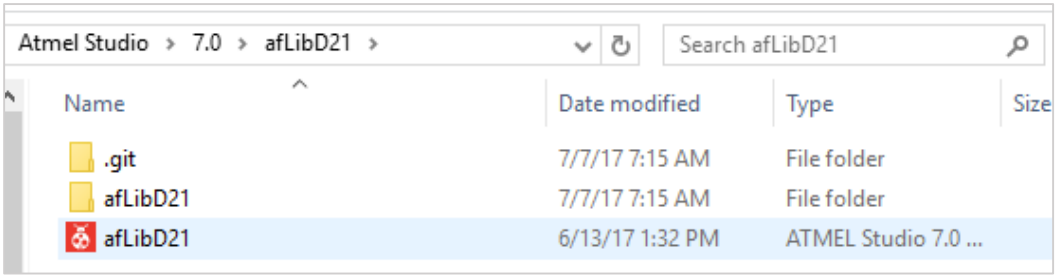
Windows 7 Users: Expand Libraries > Documents > My Documents > Atmel Studio > 7.0. Right-click the **7.0** folder and select **Paste**:



Windows 10 Users: Expand This PC > Documents > Atmel Studio > 7.0. Right-click the **7.0** folder and select **Paste**:



5 Double-click the **7.0** folder, then double-click the **afLibD21** folder to open the project's files. Double-click the **afLibD21** Atmel Studio 7.0 Solution file (with the red icon) to open the project in Atmel Studio:

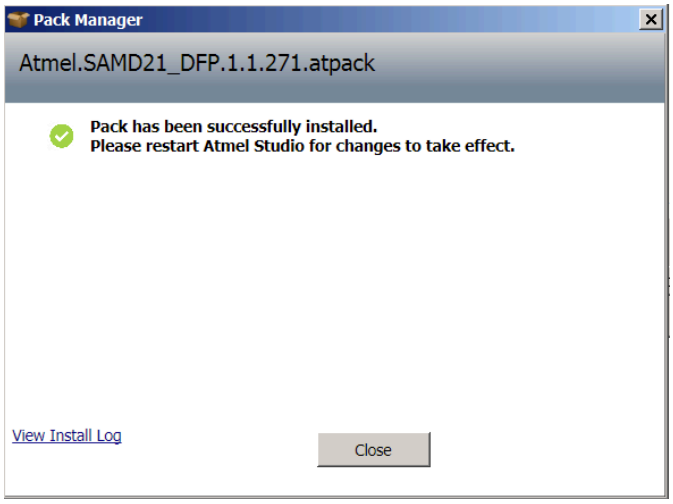


6 If you have just installed Atmel Studio, you will be prompted to install an Update Pack for the SAMD21 board; ensure the **Update to version** matches the **Missing Version**.

a. Click **Download and Install**. Installation only takes a few moments.

b. When it completes, click **Close** on the Pack Manager screen, then **Close** again on the Update Pack screen.

c. Close Atmel Studio 7 and relaunch the application by double-clicking the **afLibD21** solution file, which should still be open in the File Manager on your computer. If not, open your Documents folder, expand Atmel Studio, expand 7.0, expand the afLibD21 folder, then double-click the **afLibD21** solution file to relaunch Atmel Studio.

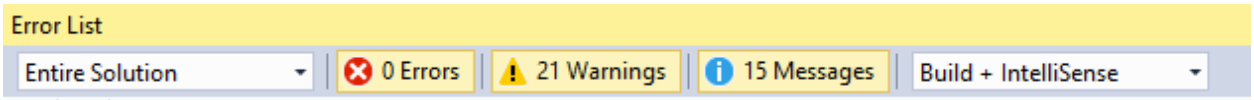


4.5 Build the Lab Project

1 After Atmel Studio relaunches, find the Solution Explorer window in the right side of the AS7 workspace. If you can't see the Solution Explorer window, close the VA View window or any other window open above it.

The Solution Explorer will list out the files in the project and let you review or modify them. Scroll to the file **main.cpp**; click to open it in the workspace so we can review how it works.

2 In the Build menu at the top of the screen, select **Build Solution** to build the project. It will take a minute or so. Ignore any warnings or messages, the solution should build with 0 errors:



4.6 Configure Your Modulo Board

We need to install a profile to your Modulo board that supports the attributes used by this project. By now, you should be pretty familiar with the Afero Profile Editor, so we'll take a shortcut here and use a preinstalled profile.

1 Minimize Atmel Studio by clicking the Minimize icon in the upper-right of the window.

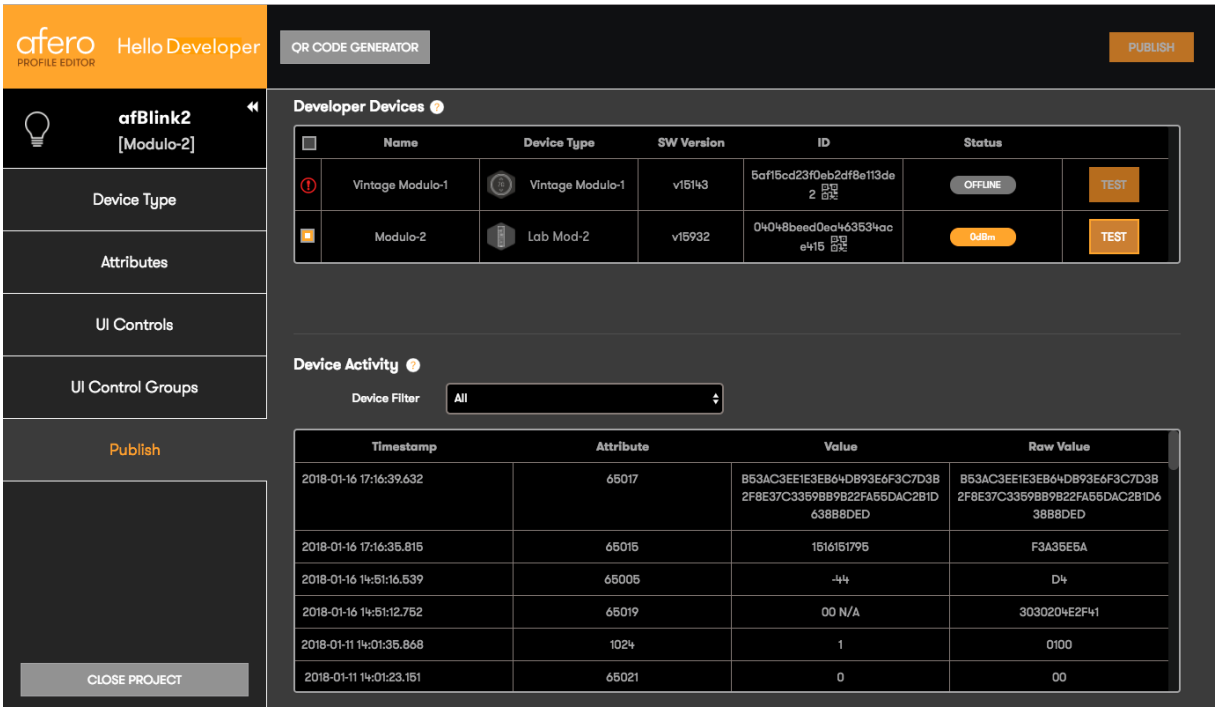
2 Launch Afero Profile Editor, sign in if requested, and select **Open** to open a local device profile.

3 In the File Explorer window, select your Documents folder, and drill down to Documents > Atmel Studio > 7.0 > afLibD21 > afLibD21 (again) > profile. In that folder you will see two other folders, **afBlink** and **afBlink2**.

For the Modulo-2 board, click once on the **afBlink2** folder and then click **Select Folder**. The profile will open.

4 Detach your Modulo board from any previous hardware it's connected to and plug it in to a USB port with a Micro-USB cable, all by itself.

5 Select **Publish** in the left-hand Navigation pane. When the Modulo comes online, publish the profile to your board. At this point you should be pretty familiar with the process; if not, please review the materials in Lab 2. Using the Afero Profile Editor.

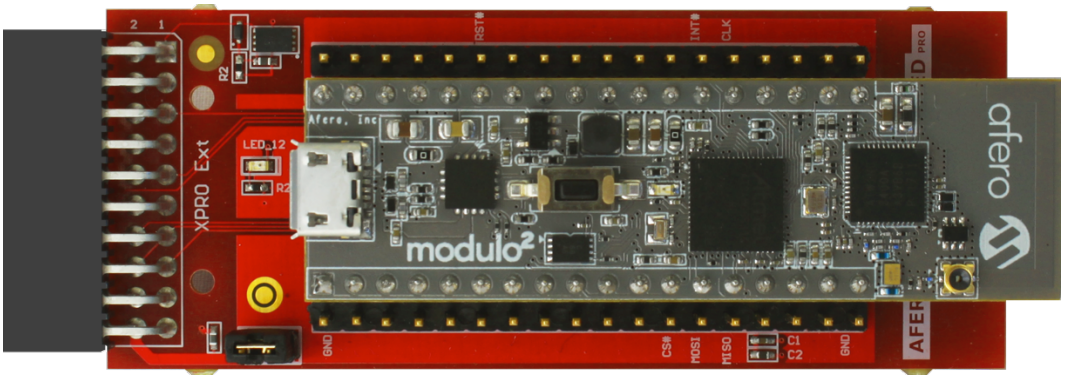


Once your Modulo has the afBlink2 profile loaded on it, continue to the next section.

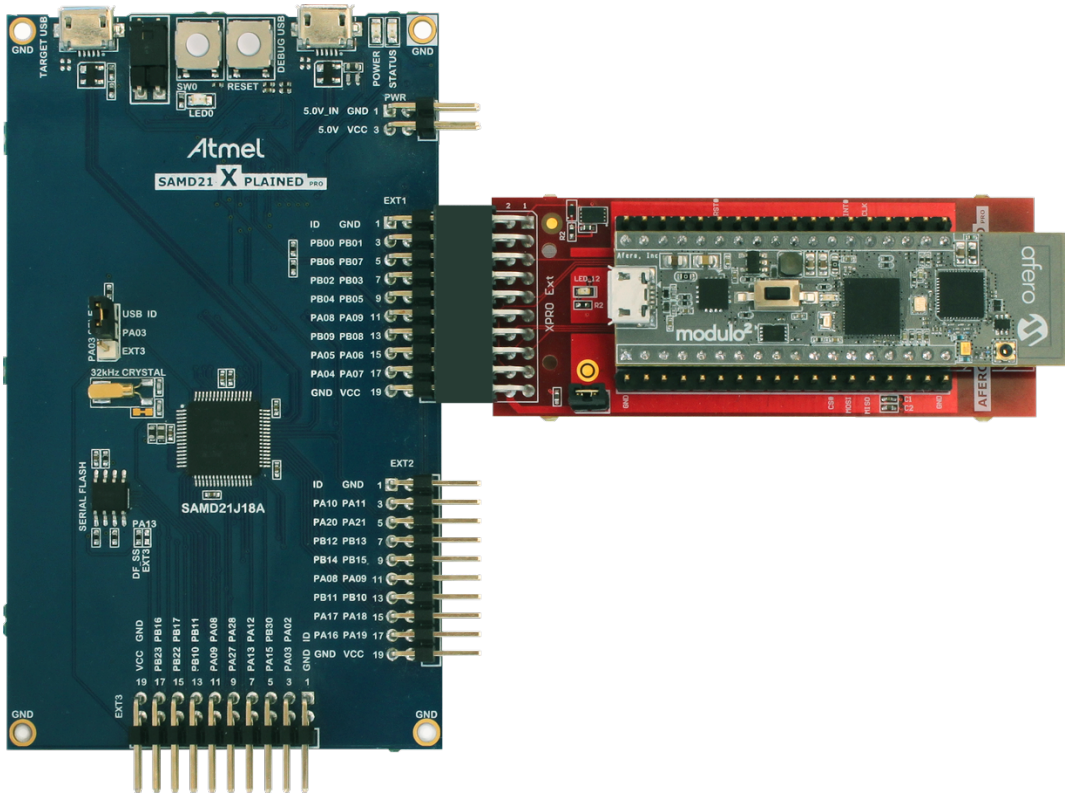
4.7 Assemble the Lab Hardware and Download the Application

1 Remove the SAMD21 XPlained Pro and the Afero-Mod2 Xplained Pro interface board from their packaging.

Unplug your Modulo board and insert it into the interface board. The USB port of the Modulo should be in the same direction as the connector of the interface board. The connection should be fairly tight, and you will have to press the board down until the connector “clicks” to seat it properly. Take care to not bend any pins on the Modulo board as you insert it.



2 Connect this interface board to the EXT1 connector of the SAMD21 board as shown:



3 Connect the SAMD21 board to your computer with a Micro-USB cable. Plug the cable into the DEBUG USB port on the SAMD21. Do not connect a cable to the TARGET USB port.

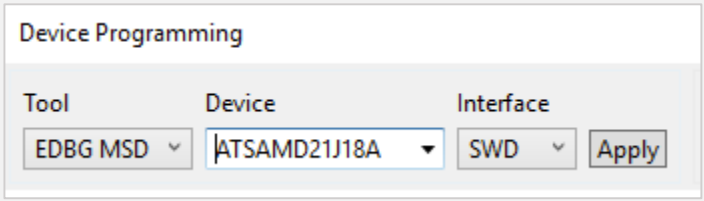
4 Windows may install device drivers for the board, which will take a few moments. If Windows Autoplay pops up to ask you what you want to do with the device, just click the **Close** button on that window.

The SAMD21 board should pop up in Atmel Studio once it's connected. The extension board may be listed as "Unknown" because it's a new board design. This is OK.

5 Download the application to the SAMD21 board. Click the **Debug** menu at the top of the window and select **Start without Debugging**. This will download the application to the SAMD21 board and start executing it.

NOTE If you get any errors connecting to the SAMD21 board, please check that the hardware debugger and board type are correct:

Click the Tools/Device Programming menu, and ensure that the EDBG tool and the SAMD21 device are selected. If they are not, select them and click **Apply**.



Then disconnect and reconnect the SAMD21 board and you should be able to continue.

4.8 Blink an LED

1 In the Afero mobile app, select your Modulo device. The profile will have a Blink menu with Off and On buttons. Tap **On** and note that the LED on the Modulo will start blinking. Tap **Off** and the LED will stop blinking.

2 Back in Atmel Studio, locate the **main.cpp** code window. If it's closed, go back to the Solution Explorer window and click **main.cpp** to open it. You can close some of the other windows in this workspace to get a larger view of the code.

What's Happening?

When you press the Blink button in the mobile UI, that attribute (BLINK) is sent to the MCU. The MCU receives this update via the `attrSetHandler` function in the code:

```
bool attrSetHandler(const uint8_t requestId, const uint16_t attributeId, const
uint16_t valueLen, const uint8_t *value) {
    printAttribute("attrSetHandler", attributeId, valueLen, value);

    switch (attributeId) {
        // This MCU attribute tells us whether we should be blinking.
        case AF_BLINK:
            blinking = (*value == 1);
            break;
```

When the BLINK attribute changes, the value passed to this code is 0 or 1 (for "Off" or "On"). This code will turn on the local Boolean variable **blinking** to tell the MCU whether it should be blinking the Modulo LED or not.

If you look in the `loop` function, you can see this in action:

```
void loop() {
      if (blinking) {
            if (millis() - lastBlink > BLINK_INTERVAL) {
                  toggleModuloLED();
                  lastBlink = millis();
            }
            } else {
            setModuloLED(false);
      }

      // Give the afLib state machine some time.
      aflib->loop();
}
```

One thing to note there is the call to `aflib->loop()` at the bottom of this function. `afLib` itself needs a little CPU time to process its queue of outgoing attribute updates (if there are any) and to check for incoming attribute changes coming from the Afero Cloud (if there are any of those). It's important to give `afLib` enough CPU time to process these tasks, so in any `idle` loop of your MCU code, please ensure that `afLib's loop()` function gets called as often as possible. If `afLib` has to work to do, this loop will return immediately.

The `toggleModuloLED` function changes the LED on the Modulo by setting the Afero attribute for the LED to on or off:

```
void toggleModuloLED() {
    setModuloLED(!moduloLEDIsOn);
}

void setModuloLED(bool on) {
    if (moduloLEDIsOn != on) {
        int16_t attrVal = on ? LED_ON : LED_OFF; // Modulo LED is active low
        while (aflib->setAttribute16(AF_MODULO_LED, attrVal) != afSUCCESS) {
            printf("Could not set LED\r\n");
              aflib->loop();
        }
        moduloLEDIsOn = on;
    }
}
```

When the MCU sets the LED attribute via the `setAttribute16` function, this attribute change is sent to the Modulo board via `afLib`, which in turn sets the LED and sends the attribute update to the Afero Cloud. As the LED on the board changes, you can see the LED attribute in the mobile app reflect the LED state as it changes.

Be sure to check out the `attrNotifyHandler` function as well; this function is called when the MCU changes an attribute or when the Modulo sends a default attribute value to the MCU. In general, you will get an `attrNotifyHandler` event any time an attribute value changes that the MCU doesn't know about. If an attribute is specifically set by the Afero Could, you will also get an `attrSetHandler` event

so that remote attribute changes can be handled differently if needed. There is a lot of overlap in functionality between these two handlers and for some simple MCU applications it may only be necessary to use one or the other.