**Hackathon presented by AT&T and 2lemetry**

**Overview**

In this hackathon you will connect an Arduino Leonardo device to a cloud platform in order to transmit sensor values and receive commands. As part of this hackathon you will also interact with the cloud platform via a REST API to query the sensor values published to the platform.

**Quick Start**

This section provides a streamlined set of steps to get running quickly. The sections that follow this one go into more detail on each of these steps.

1. Setup Arduino IDE  
   The Arduino IDE is found at: <http://arduino.cc/en/main/software>
2. Download library.

The library can be found here: https://github.com/attM2Mfoundry/hackathon-09-14-2013/tree/master/Arduino3G. Download and [setup](#id.icgxom9pgyqi) in Arduino IDE.

1. Copy example sketch.

The library includes a couple examples. Copy one of these into a New Sketch. The easiest to start with is the sensor loop. Don’t worry at this time about the circuitry as the sketch will send whatever value is read on the analog pin. We’re more concerned with connectivity at this point rather than accurate sensor values.

1. Connect Libelium 3G shield to Arduino Leonardo.

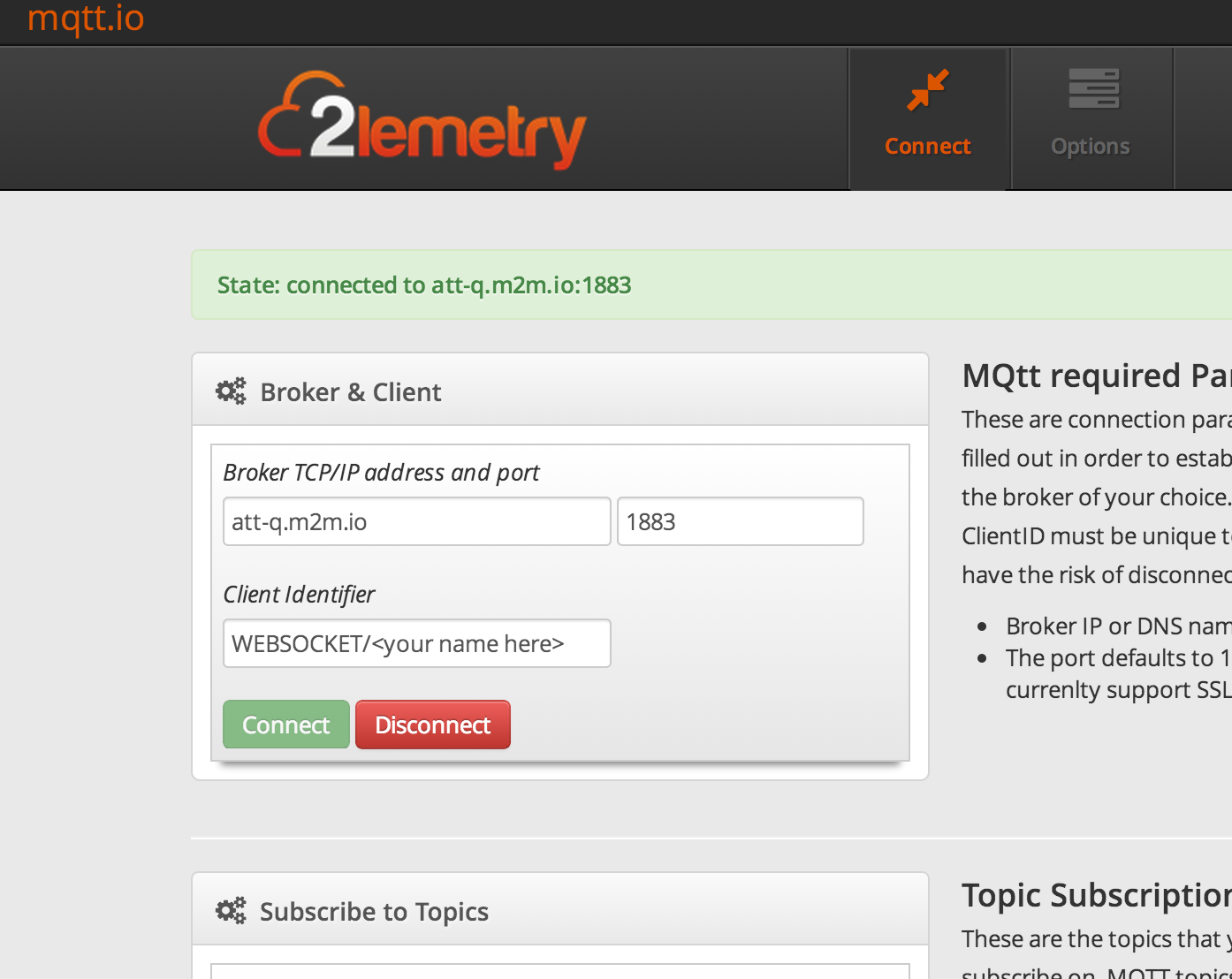
The 3G shield should include an arena. This is highly recommended for 3G connectivity. Snap the shield on top of the Arduino Leonardo matching pins to connect the two.

1. Connect USB from host computer to Arduino Leonardo (not shield).

Standard step for Arduino development. The USB cable allows for uploading sketches to the Arduino as well as outputting printed debug/information statements.

1. Connect MQTT client to observe real-time data.

The ATTCloudClient library utilizes MQTT messaging underneath to communicate real-time data with the AT&T message broker. To observe this data while it is being published, connect an MQTT client. A web-based client can be found at <http://mqtt.io>.



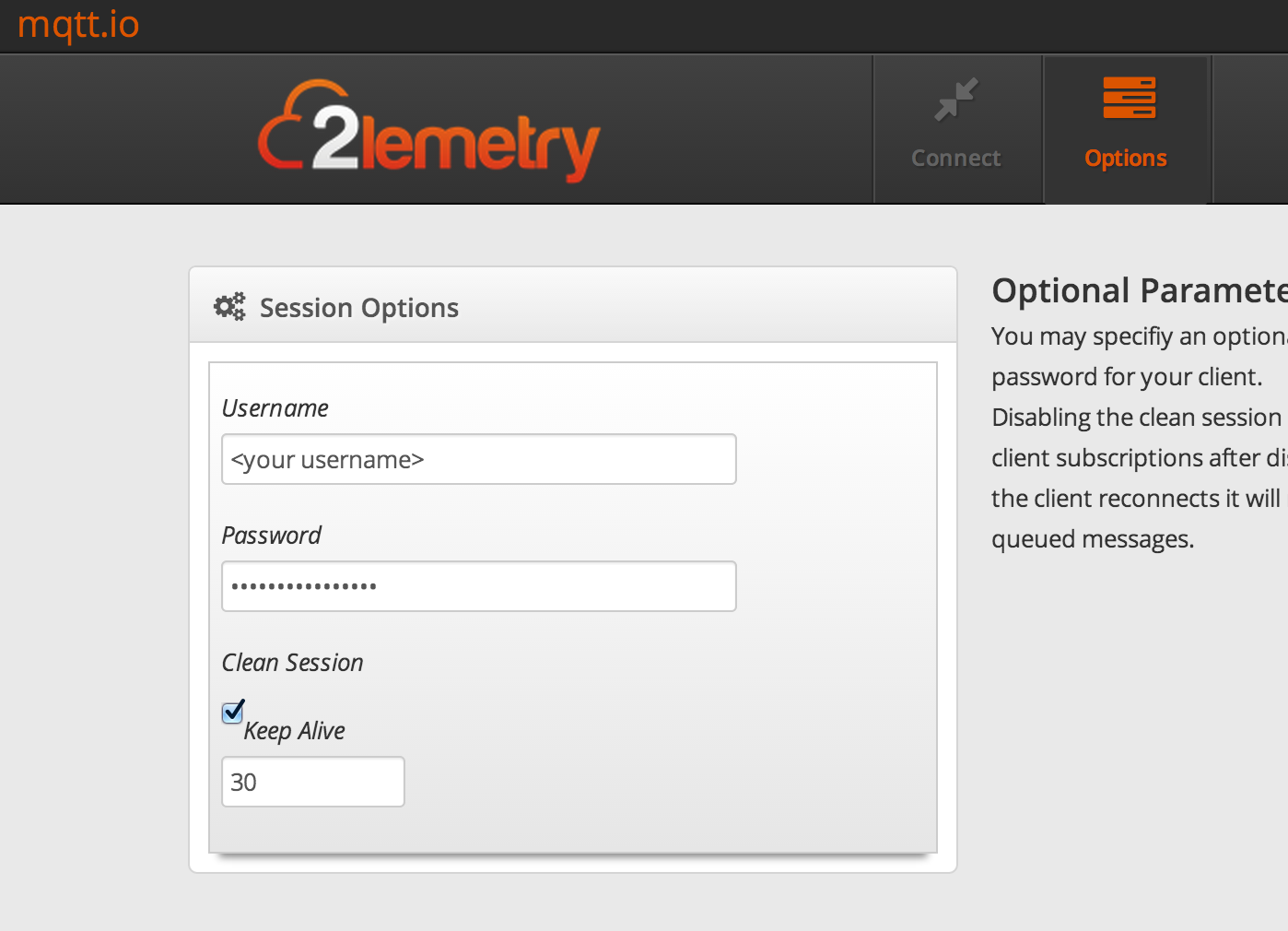
The MQTT parameters are as follows. Note the username/password are found on the Options tab. These need to be populated before clicking on Connect.

Server: att-q.m2m.io

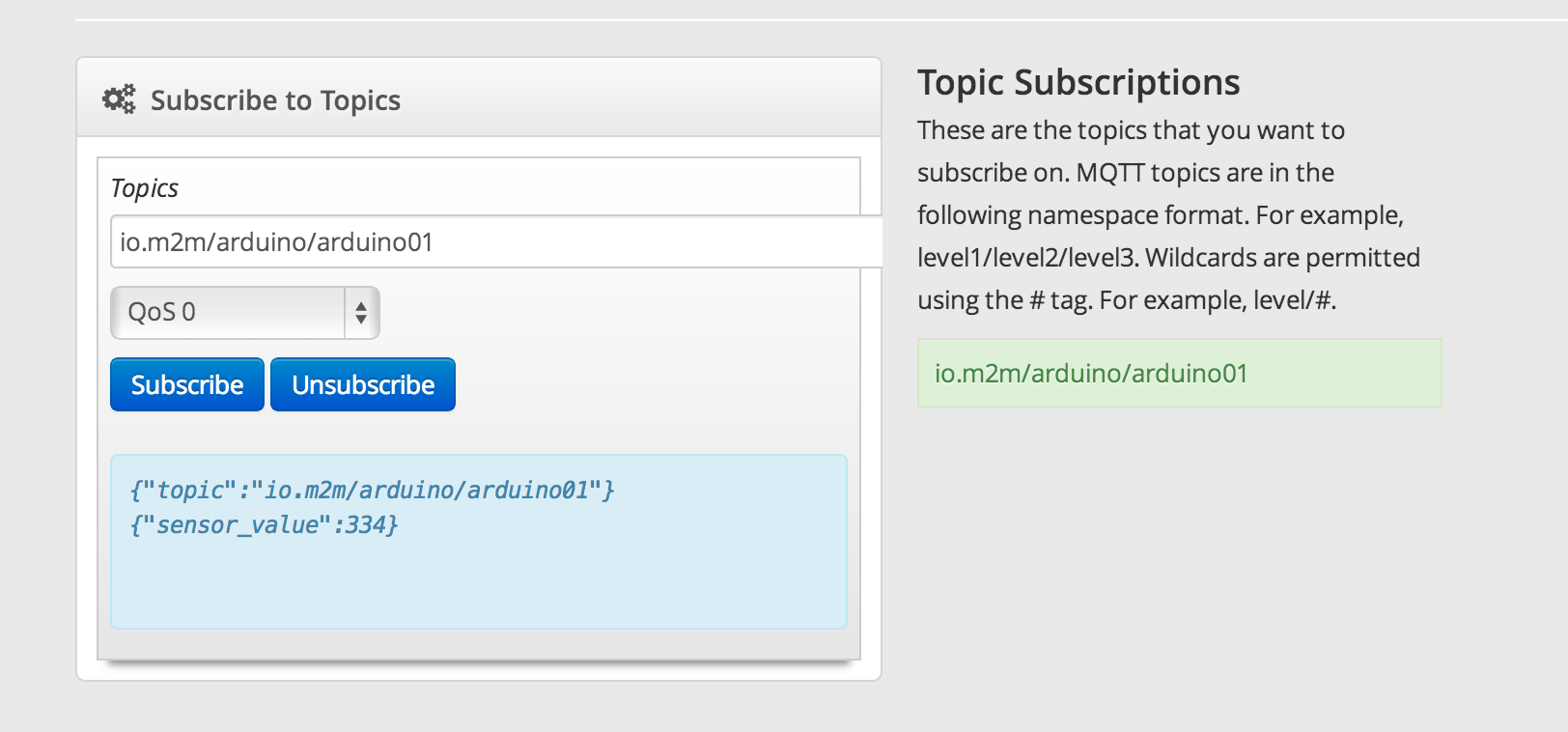
Port: 1883

Username: Platform username given in the hackathon.

Password: Password assigned at time of the hackathon.



Once connected, subscribe to the topic <domain>/<device type>/<device ID>. (io.m2m/arduino/arduino01 in this example)



Note: In the portal the term *stuff* is used in place of device type. Likewise the term *thing* is used in the place of device ID.

1. Populate the constants at the beginning of the sketch.

There are a number of constants at the beginning of the sketch. Update them as follows:

USERNAME: <username on AT&T platform>

PASSWORD: <md5 sum of clear text password>

DOMAIN: <Your assigned domain> This can easily be found on the Account tab in the portal.

DEVICE\_TYPE: This is an arbitrary string to classify devices into different capability groups.

DEVICE\_ID: The unique device in this experimentation / hackathon project.

1. Build & Upload Sketch

Press the Upload button to build and upload the code to the Arduino. When using the Arduino IDE to upload it has been observed that a “serial port in use” error occurs. This sometimes can be eliminated by reselecting the serial port by selecting Tools > Serial Port and then clicking on the proper port. This may already be checked with a check mark. Select it anyway.

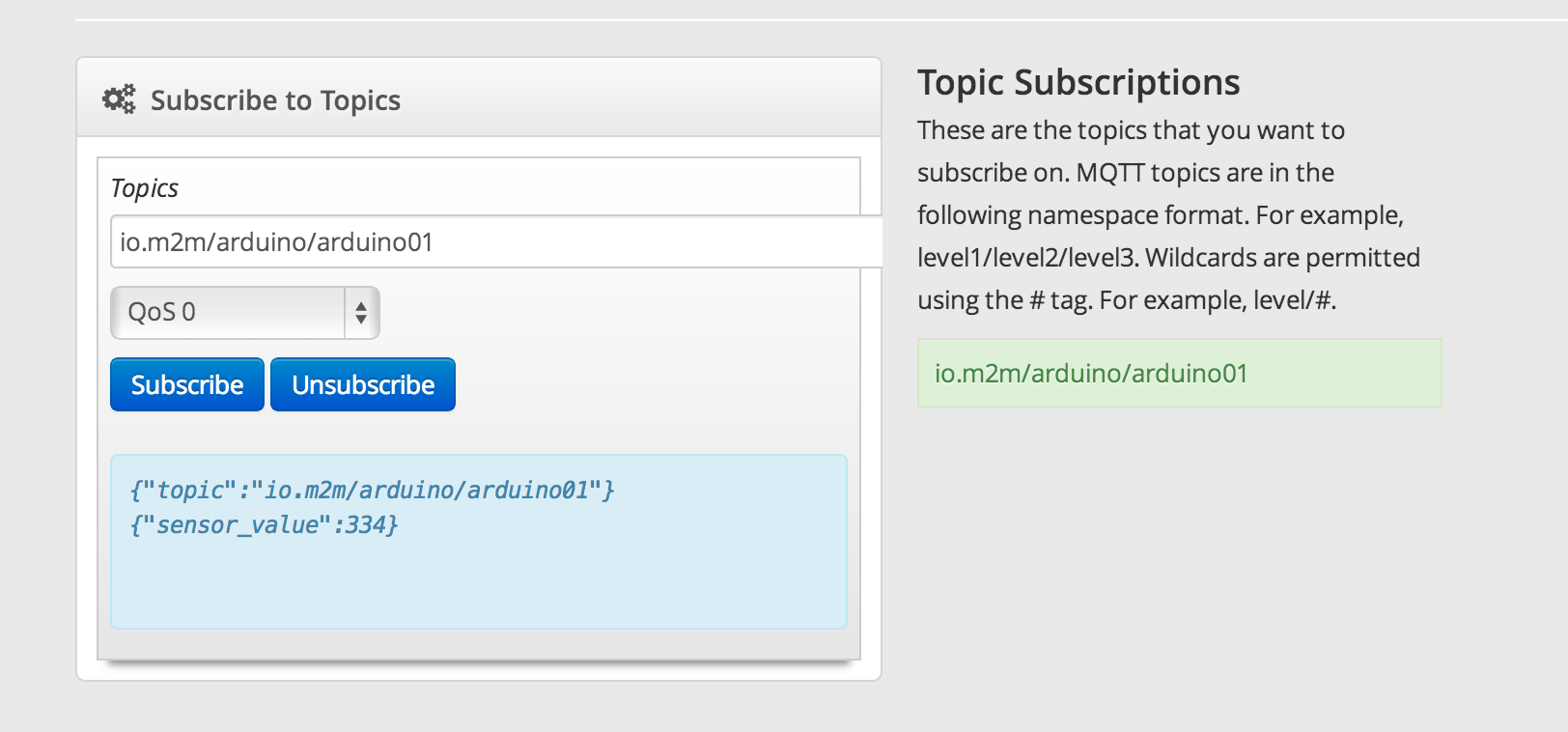
1. Open serial monitor

Expected successful behavior is similar to: update with any library changes

AT  
START  
+STIN: 25  
+CPIN: READY  
AT  
ATOK  
+VOICEMAIL: INIT\_STATE, 0, 0  
OK  
Connecting to the network...

Connected to network  
AT+CGSOCKCONT=1,"IP","m2m.com.attz"  
AT+CGSOCKCONT=1,"IP","m2m.com.attz"OK  
AT+NETOPEN="TCP"  
AT+NETOPEN="TCP"Network opened  
Network opened  
AT+TCPCONNECT="att-q.m2m.io",1883  
AT+TCPCONNECT="att-q.m2m.io",1883Connect ok  
AT+TCPWRITE=99  
AT+TCPWRITE=99  
Loop...  
::read timeout  
Loop...  
::read timeout  
Loop...

1. Observe real-time data. In the MQTT client you will see sensor values published:



1. Power Down

Remove USB cable from host computer to power down Arduino and 3G shield.

1. Observe past data in portal
2. Make a past API call to retrieve same data

Below is an example of using the command-line client cURL to query data previously published to the platform:

curl –user <user name>:<password>

<http://att-api.m2m.io/2/account/domain/mydomain/stuff/mythings/thing/thing001/past?attributes=_>

And an example response:

{

"domain": "mydomain",

"stuff": "mythings",

"thing": "thing001",

"lastentry": "1369518276118890",

"results": {

"1369592198946353": {

"sensor\_value": 100

},

"1369592187210084": {

"sensor\_value": 101

},

"1369592182612328": {

"sensor\_value": 99

},

"1369590782704041": {

"sensor\_value": 22

},

"1369590779565053": {

"sensor\_value": 23

},

"1369518276118890": {

"sensor\_value": 25

}

}

}

**AT&T Arduino Library**

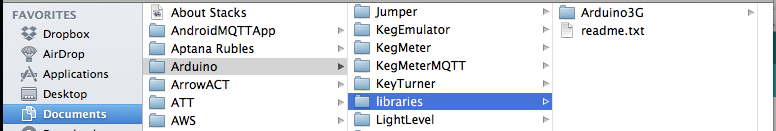
An Arduino library has been created for this hackathon that allows easy connection of the Arduino to the cloud. This library abstracts the underlying messaging protocol and allows easy publishing of a variety of key/value pairs.

Setup of Library

The library repo can be cloned from here: <https://github.com/attM2Mfoundry/hackathon-09-14-2013/tree/master/Arduino3G>

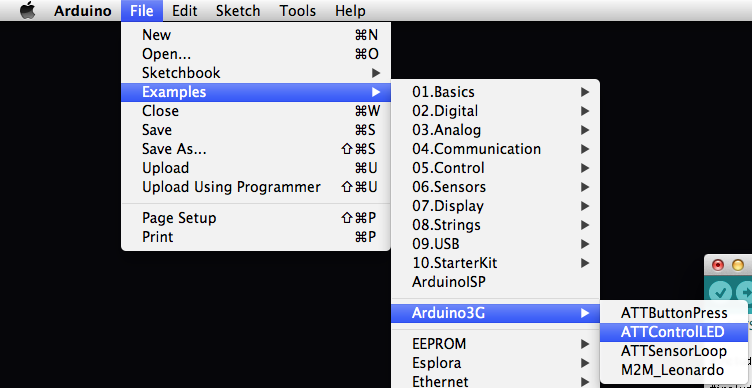
Instructions to setup an Arduino library can be found at <http://arduino.cc/en/Guide/Libraries>.

Location of libraries (Arduino IDE 1.0.1 and greater) is in the Documents/MyDocuments folder:



Examples Included in Library

The library comes with a few examples to help you get started quickly. These can be found through the Arduino IDE by selecting: File > Examples > ATTCloudClient:



Including library in sketch:



The following is an overview of a few of the key methods provided by the ATTCloudClient.

The following references are required. The first is the CloudClient object which provides the messaging functionality. The second is the driver interface for the Arduino 3G shield.



The following two lines instantiate the client and perform the connection to the cloud server. More information on the constants in these methods can be found below. When instantiating the client, a callback method is given. This method is defined in the user’s sketch and handles processing of commands received from the cloud server.



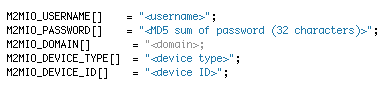
This method configures the cloud client to use the user’s messaging space (domain). More information on the constants can be found below.



The following method sends a key-value pair to the cloud store. There are many methods for sending values which can be found in the ATTCloudClient reference provided.



At the beginning of the example sketches you will see a series of constants which configure things such as username and password for the cloud server.



USERNAME: The username on the AT&T M2M platform. This will be supplied to you as part of the hackathon materials.

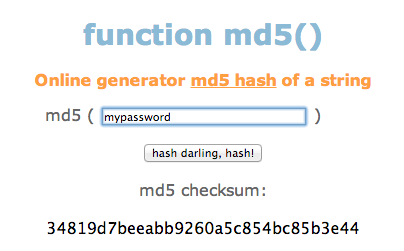
PASSWORD: The password on the AT&T M2M platform. This will be supplied to you as part of the hackathon materials. When using it in the Arduino sketch, the password is the MD5 hash of the clear-text password. This will be a 32 character value. MD5 hashes can be found by running local tools:

OSX:



Linux:



Or by using an online MD5 hash generator at: http://www.md5.cz. 

The messaging space on the AT&T cloud platform is separated by three parameters: domain, device type and device ID. Each account has an associated domain. Under that domain there can be many device types and likewise under each device type many devices can be given. A simple example may help illustrate this concept.

Suppose you have around 100 devices which measure temperature and report the values back to the platform. Some are Arduinos, some are Raspberry Pis and some are PCs. All off these devices would report on the same domain. All of the Arduinos would report on their device type under their respective device ID.

Domain / Device Type / Device ID

<domain> / arduinos / arduino01

<domain> / arduinos / arduino02

<domain> / arduinos / arduino02

<domain> / pc / pc01

<domain> / pc / pc02

<domain> / pc / pc03

<domain> / pc / pc04

DOMAIN: The user’s domain on the AT&T platform.

DEVICE TYPE: The type of device. This is an arbitrary field to separate devices into smaller buckets by type. For this hackathon using the value “arduino” makes sense.

DEVICE ID: A unique name for each device reporting. This can be something arbitrary such as “arduino01” as long as it is unique across your domain.

Sending Values to the Cloud Platform

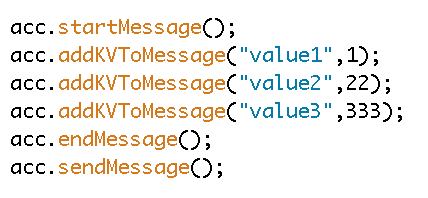
The messaging protocol that underpins the ATTCloudClient is MQTT. MQTT provides a simple publish/subscribe architecture. For the most part in this hackathon the underlying MQTT protocol is abstracted away and in its place methods are provided to send key-value pairs to the platform.

The following two examples show two ways to send values to the platform. The first involves sending one key-value pair:



This sends the key “sensor\_val” and its respective value, the integer value stored in sensorValue. This example would be useful if the device implemented a single sensor and only needed to send a periodic update of the value.

Another example involves building a message containing multiple key-value pairs:

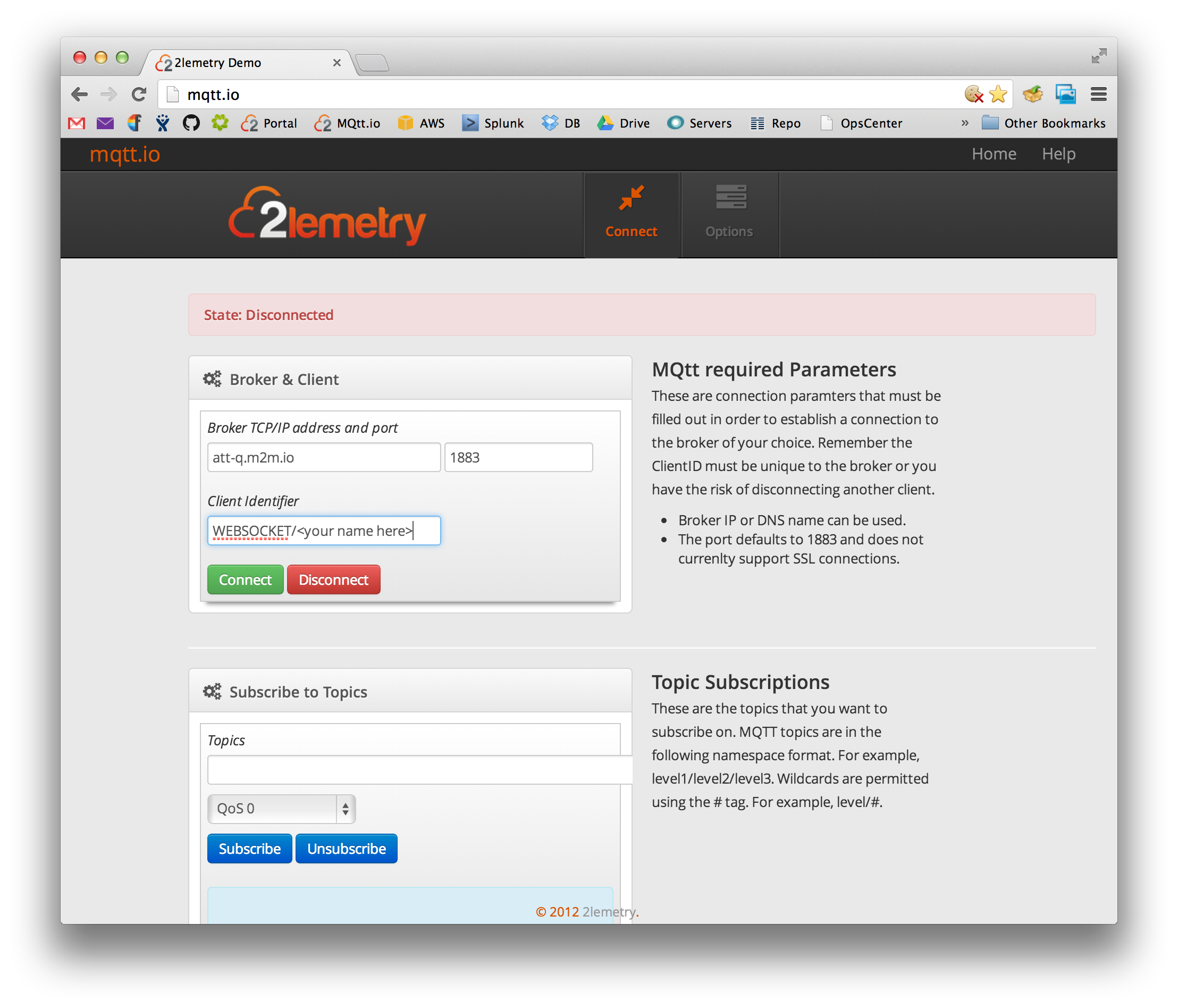


The message is created and sent by calling startMessage, adding KV pairs, ending the message by calling endMessage and then sending the message. As the message being created is built inside the library this series of calls must be executed in the above order. From 1 to 5 K-V pairs can be added to a message.

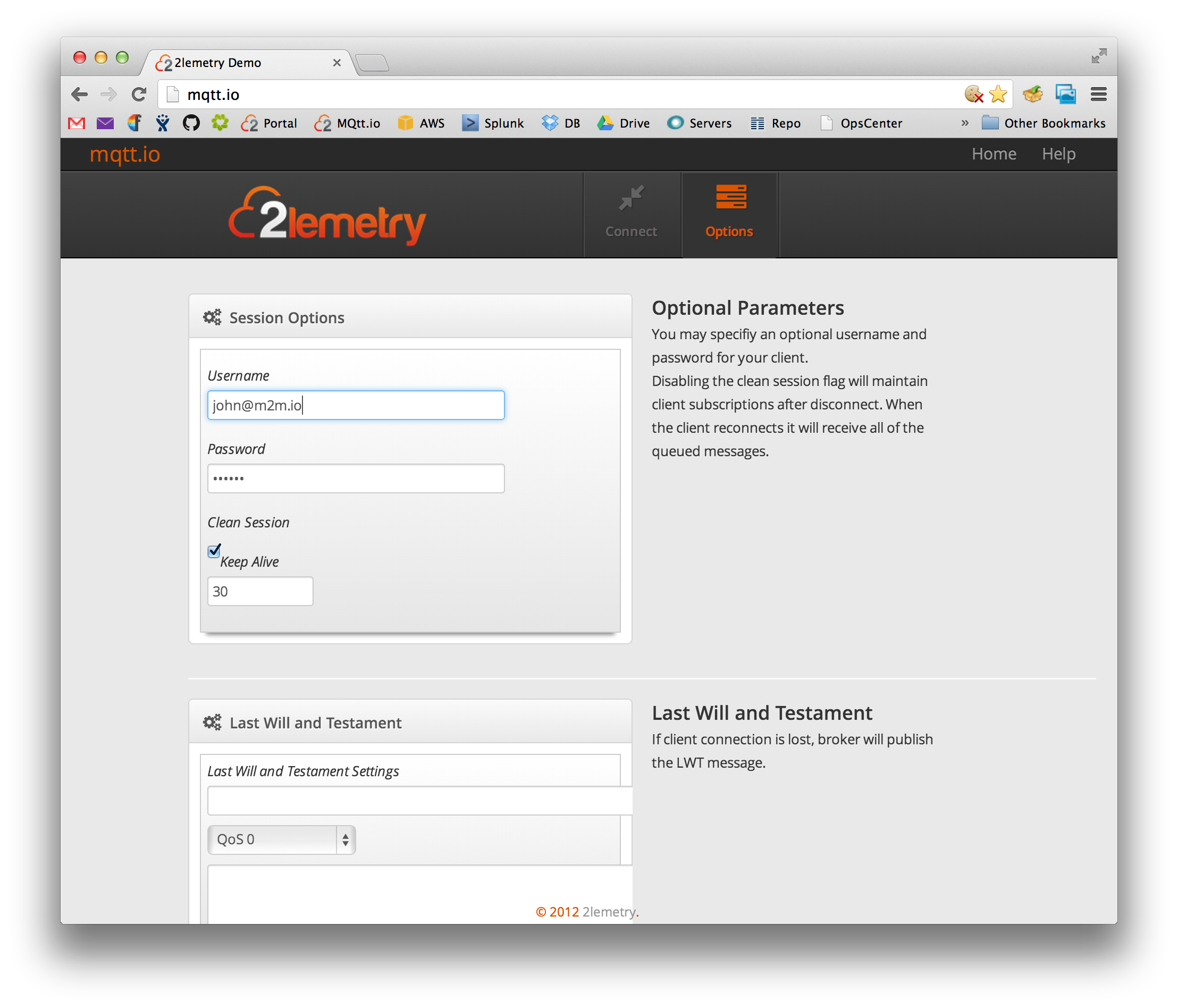
Sending Commands to the Device from the Cloud Platform

The ATTCloudClient library also provides the ability to register to receive commands from the cloud platform. The ATTControlLED example included with the library shows receiving a “light on” / “light off” command and turning on or off an LED according to the command received.

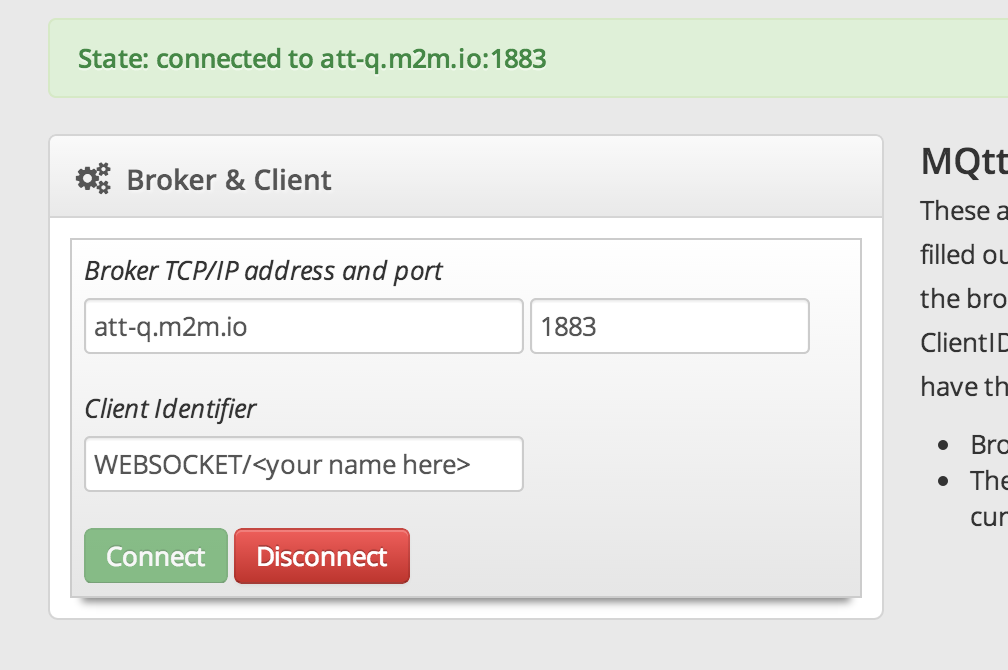
To send a command from the cloud platform, navigate to <http://mqtt.io>. Login by using the following parameters (the same as real-time above):



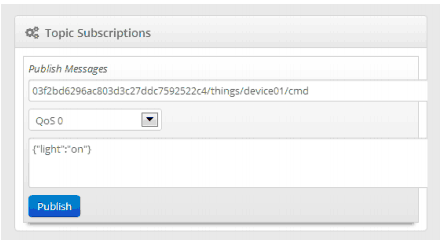
Click on the Options tab to input your username / password. Do this before clicking on Connect.



After entering your credentials, click on Connect. You should see the green Connected bar appear:



Every device is setup by the library to publish on the MQTT topic <domain>/<device type>/<device ID>. For commands a /cmd topic level is appended: <domain>/<device type>/<device ID>/cmd. To publish a command to the device, use “Topic Publish”.



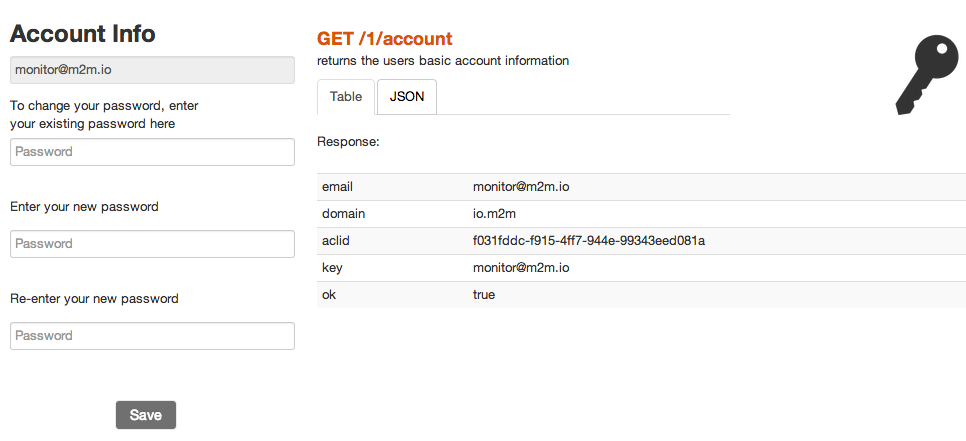
**API**

The REST API can be found hosted at the following URL: <http://att-api.m2m.io>. Note this is the base URL for the API, and cannot be accessed as a regular web site. Detailed documentation on API calls can be found in the API reference document supplied in the hackathon documentation.

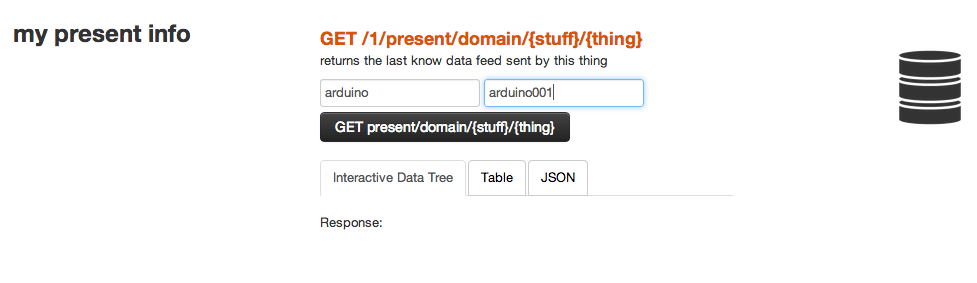
**Portal**

There is a portal that allows a graphical interface to the REST API. It can be found at: <http://att.m2m.io>. To login to the portal use your username and password (not MD5) for the platform.

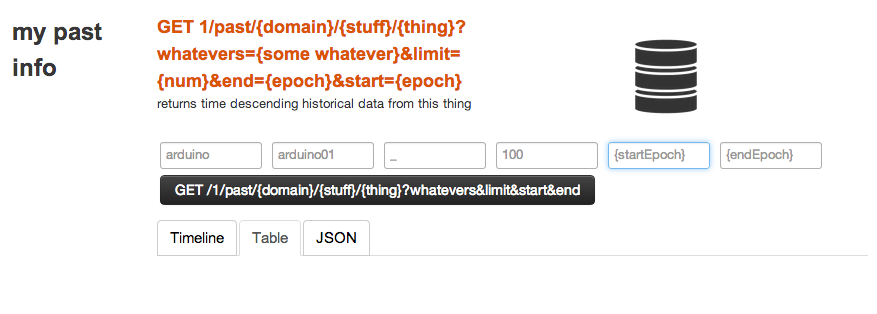
Account - The account tab provides information about your platform account. This is where your domain can be looked up and copied.



Present - A present call returns the most recently published value. This is a good way to retrieve the current value of a sensor when building an application. When the desired query parameters have been populated, click on the black button (w/API reference) to execute API call.



Past - A past call is used to lookup past sensor values published to the platform. In the portal stuff = device type and thing = device ID. In the whatevers field, an \_ is a wildcard for all values. When the desired query parameters have been populated, click on the black button (w/API reference) to execute API call.



Note: Data will not appear in the portal until after the device has been connected and publishing on a previous occasion.

Note: Registered Device tab in portal is unused in this hackathon scenario.

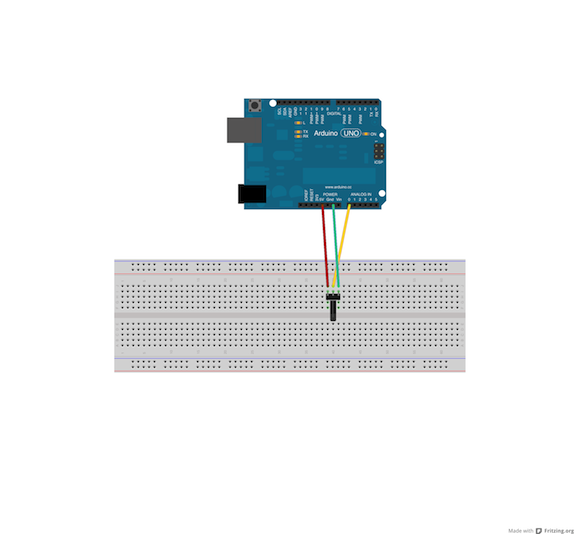
**Examples**

There are two example sketches provided with the ATTCloudClient. The following section shows the circuits used in these examples.

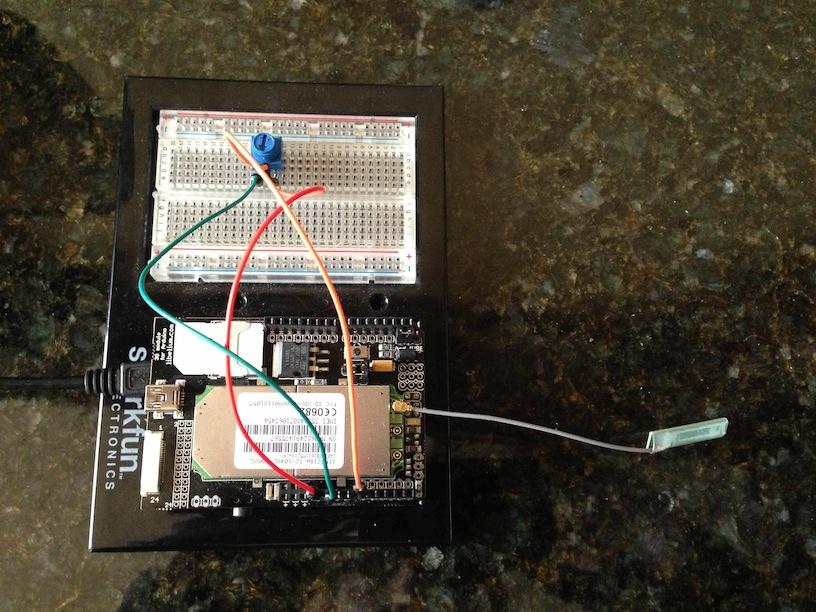
Sensor Loop

In this example a sensor value is repeatedly read and published to the platform. A potentiometer is used to simulate a sensor whose output is voltage.

Schematic



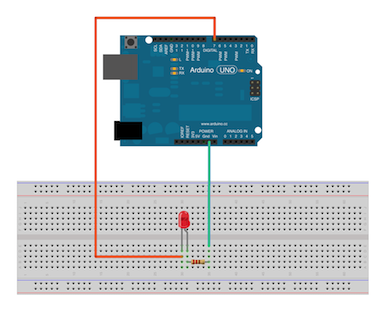
Picture



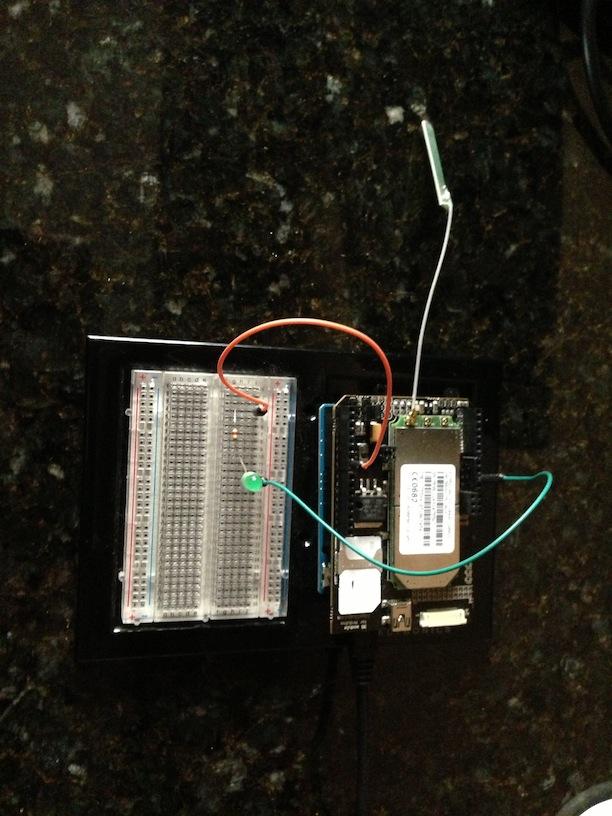
LED

This example shows the device responding to commands from the server. It will turn on/off the LED based on the content of the “light” command.

Schematic



Picture



Useful Links:

More information on the 3G Shield, including examples for HTTP, TCP, etc.:

<http://www.cooking-hacks.com/index.php/documentation/tutorials/arduino-3g-gprs-gsm-gps>

Curl utility – useful for testing REST from command line:

<http://curl.haxx.se/dlwiz/>