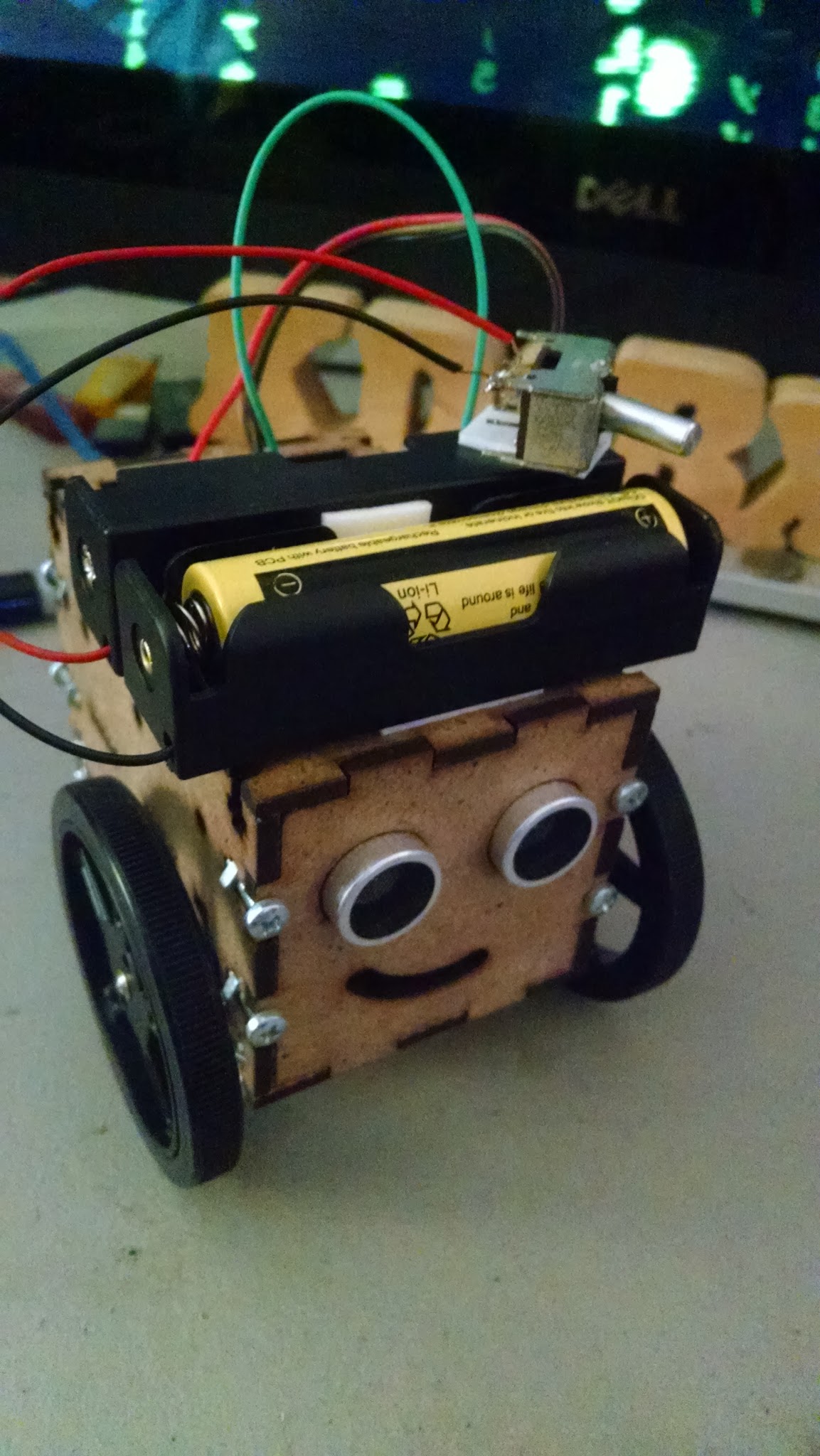
Poppet Robot with ESP8266 (NodeMCU) and Node Motor drive shield.



Design Files where found here:

<https://www.thingiverse.com/thing:1634750>

Mine can be found here:

<https://www.thingiverse.com/make:325934>

This was laser cut on a Full Spectrum Pro2416 laser, using ⅛” MDF wood with a speed set to 50% and the laser power set at 40%

I didn’t find build instructions, (so some guess work was used, and later changed)

I used M3 .50x10 screws, with M3 .50 hex nuts to secure the body together

I think the UNO board will mount with M2 screws/standoffs

From what else I can tell, this uses 9G continuous rotation servos (on order), a UNO board, Ultrasonic (HC-SR04), probably a servo shield for the Arduino.

not sure on power yet - perhaps a LIPO battery (looks like there is room for a flat battery under the UNO shelve), maybe a LIPO charger.

What I ended up changing: For this build I am using a ESP8266/NodeMCU board, with the motor driver shield. 2 16850 Batteries, a buck/boost regulator set for about 6volts.

I changed the servos, to DC motors in servos cases. (I found them at Adafruit)

<https://www.adafruit.com/product/2941>

And the wheels:

<https://www.adafruit.com/product/2744>

I started with using the Compact Rover Code with chances for the ESP8266.

Turns out the ESP8266 max PWM is 1023 (not 256) which ended up causing me some issues until I realized it.

<http://esp8266.github.io/Arduino/versions/2.0.0/doc/reference.html>

## **Analog output**

analogWrite(pin, value) enables software PWM on the given pin. PWM may be used on pins 0 to 16. Call analogWrite(pin, 0) to disable PWM on the pin. value may be in range from 0 to PWMRANGE, which is equal to 1023 by default. PWM range may be changed by calling analogWriteRange(new\_range).

PWM frequency is 1kHz by default. Call analogWriteFreq(new\_frequency) to change the frequency.

\*Next Steps Control the PopPet using MQTT,

Write a library for the ESP8266/NodeMCU motor driver \* Updated July 11, 2017

Some MQTT history:

<http://www.hivemq.com/blog/how-to-get-started-with-mqtt>

Using MQTT with the PopPet. The PopPet Robot will use “test.mosquitto.org” as the MQTT broker, it will publish to the topic “robotout”, and listen to the topic “robotin”.

The robot will Publish - it’s status (moving/stopped etc), and the ultrasonic distance. The robot will stop on it’s own and wait for another MQTT command should it get too close to an object.

The motors are fast.

Control Small robot using MQTT.

Control Codes:

0 Stop

1 Forward

2 Backward

3 Right Forward (left motor on)

4 Left Forward (right motor on)

5 Right Backward (left motor on)

6 Left Backward (right motor on)

Because of slowness of MQTT robot will stop itself if object is in

the way.

This is a experiment using MQTT. Currently using a public broker,

setting can be changed below.

Libraries required:

ESP8266Wifi, PubSubClient, TimedAction

Sketch Name: mqtt\_ultrasonic\_timedaction\_with\_drive

Code: <https://github.com/kd8bxp/PopPet-MQTT-Robot-example>

Linux/CLI MQTT control:

The robot can be controlled using the CLI if you open two terminals you can set one to subscribe to the topic “robotout”, and the other to publish to the topic “robotin”.

Controlling the robot using this method is not fun, but can be done. It is however a good way to watch the status of the robot.

To subscribe:

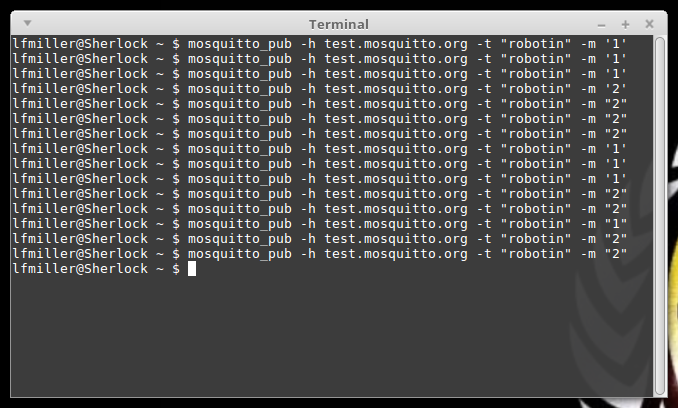
mosquitto\_sub -h test.mosquitto.org -t “robotout”

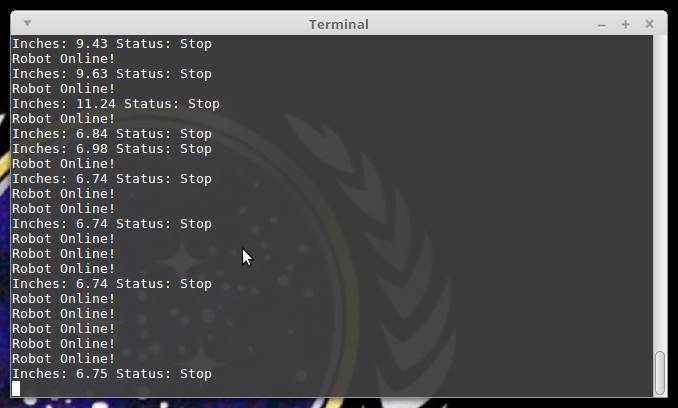
To publish:

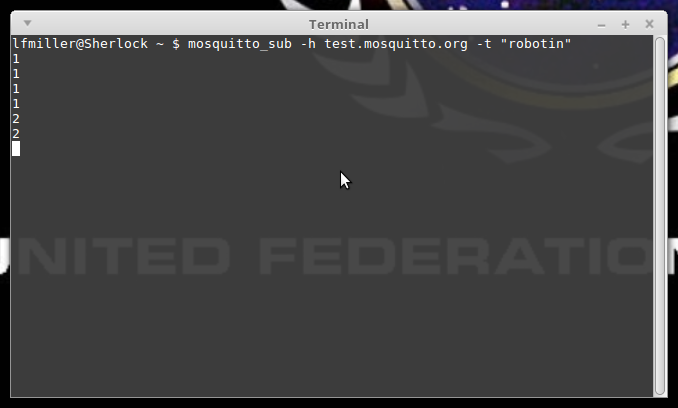
Mosquitto\_pub -ht test.mosquitto.org -t “robotin” -m ‘ {Command} ‘

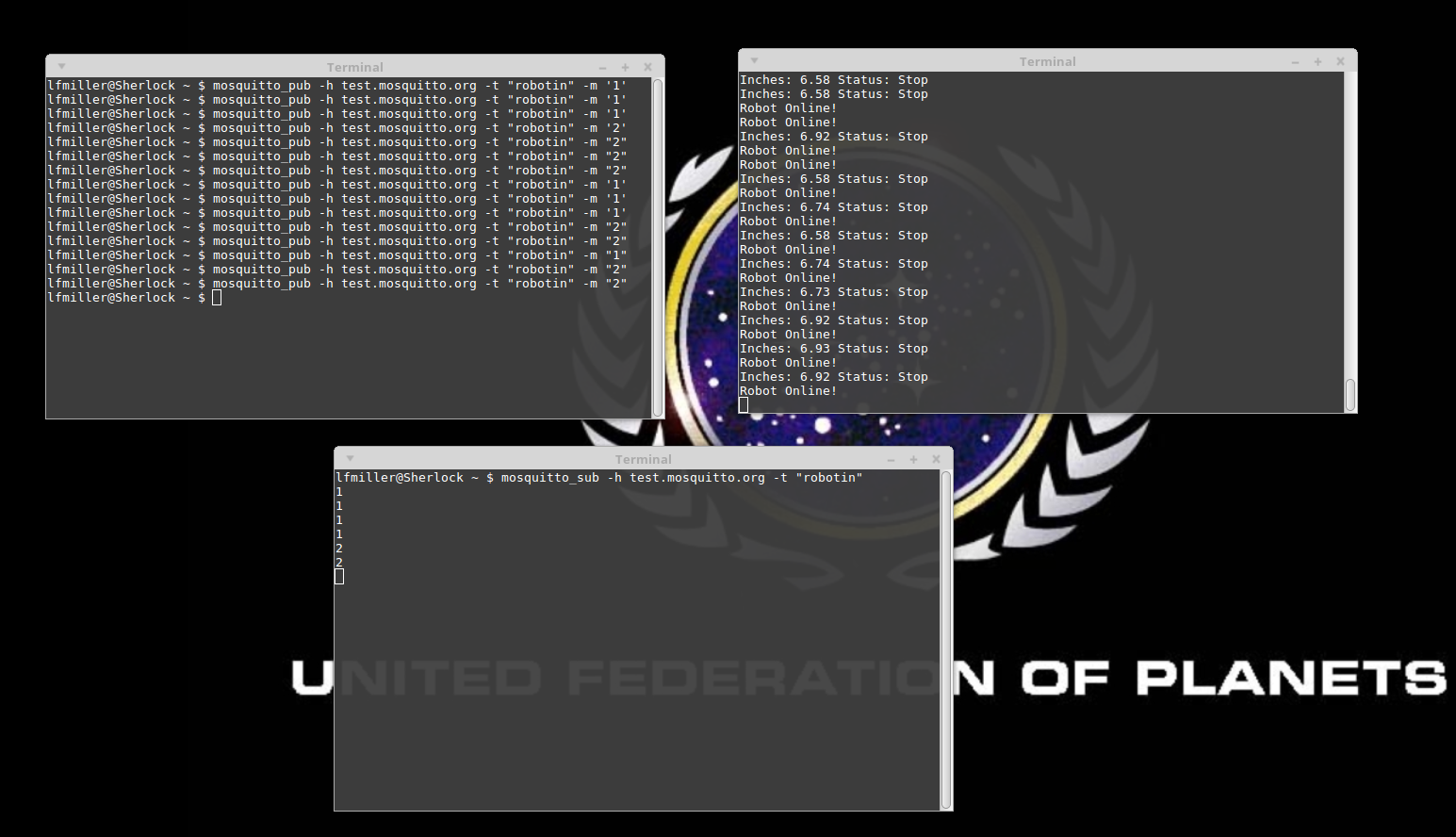
The {Command} is a number from 0 to 6 referenced above.

You can also subscribe to your robotin topic - just for verification.









So is there a Better interface?

That was the question I had - and I did some searching for using MQTT over the web.

Turns out it’s not as easy as it sounds, you need to use websockets, and there are a lot of libraries and examples for doing just that. Unfortunately I didn’t get any of them working the way I wanted. \*\* They claim this is dead simple to do….. \*\*\*

\*\* These are web interfaces, but have the same problem as using the CLI \*\*

<http://www.hivemq.com/demos/websocket-client/>

<http://www.hivemq.com/blog/full-featured-mqtt-client-browser>

\*\* I got none of these working \*\*

<http://mitsuruog.github.io/what-mqtt/>

<https://github.com/mcollina/mosca/wiki/MQTT-over-Websockets>

<https://www.npmjs.com/package/mqtt>

What is needed is a dashboard that allows you to design your own buttons and interface.

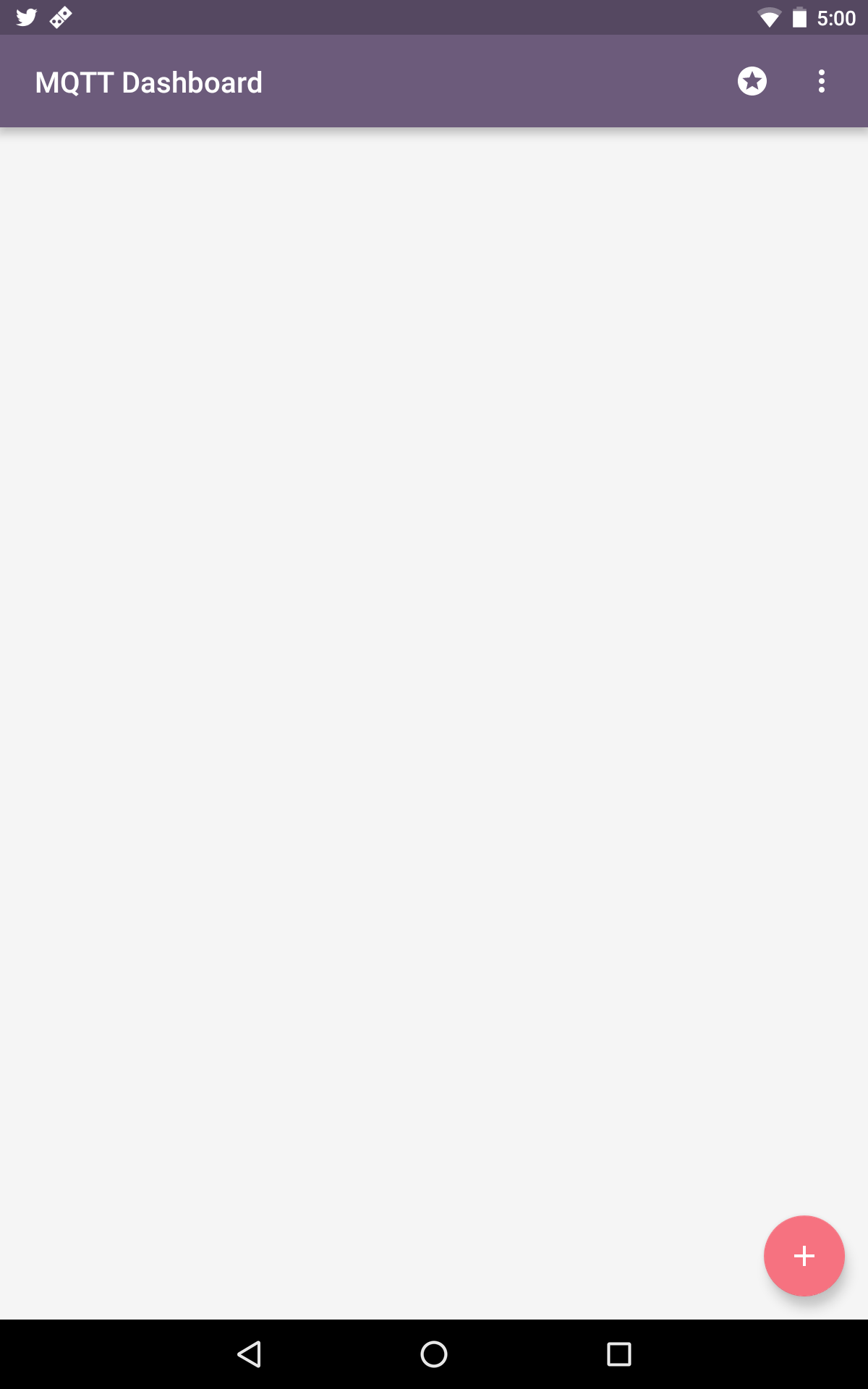
What I found was a mobile app that would allow you to do this.

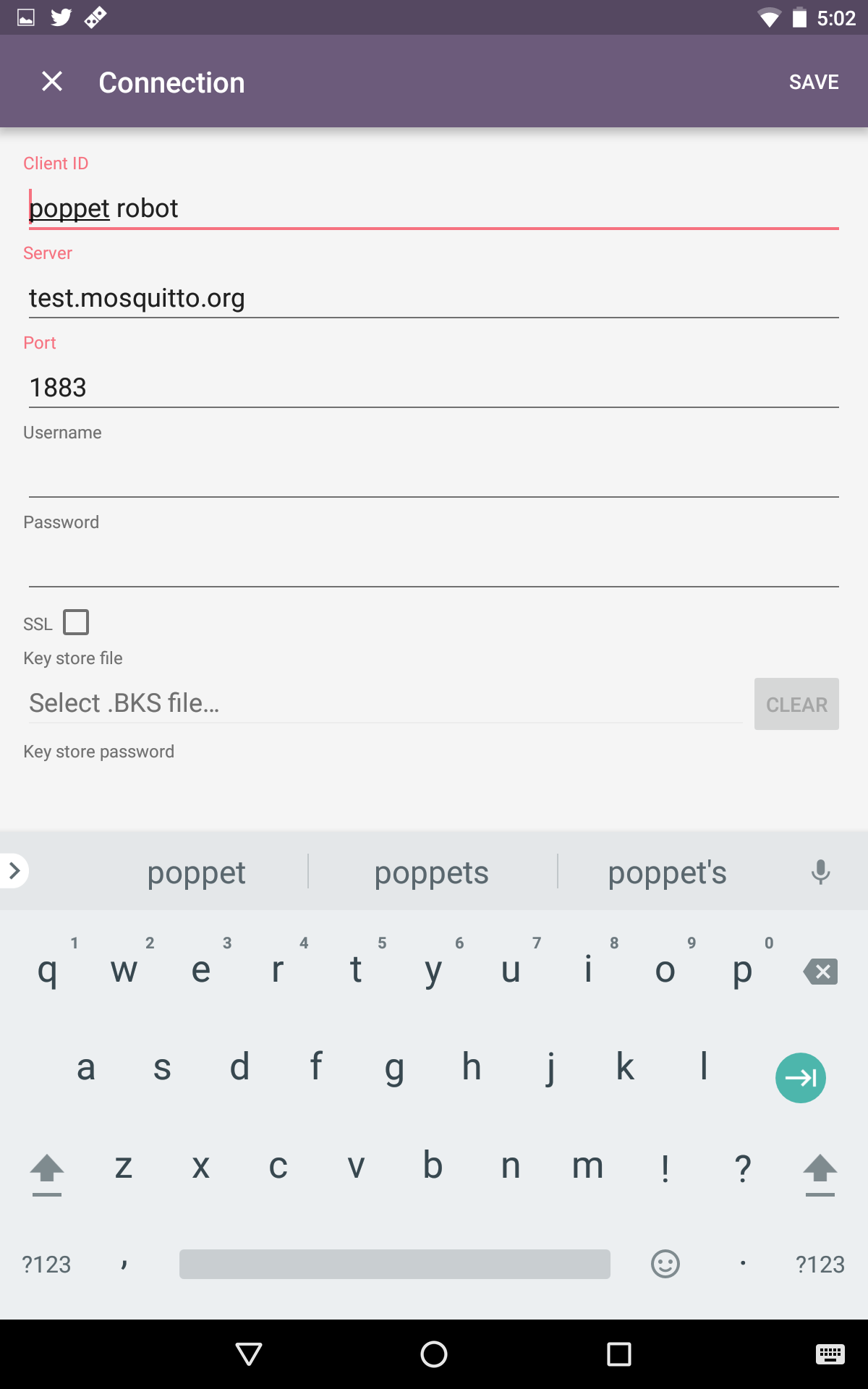
<https://play.google.com/store/apps/details?id=com.thn.iotmqttdashboard&hl=en>

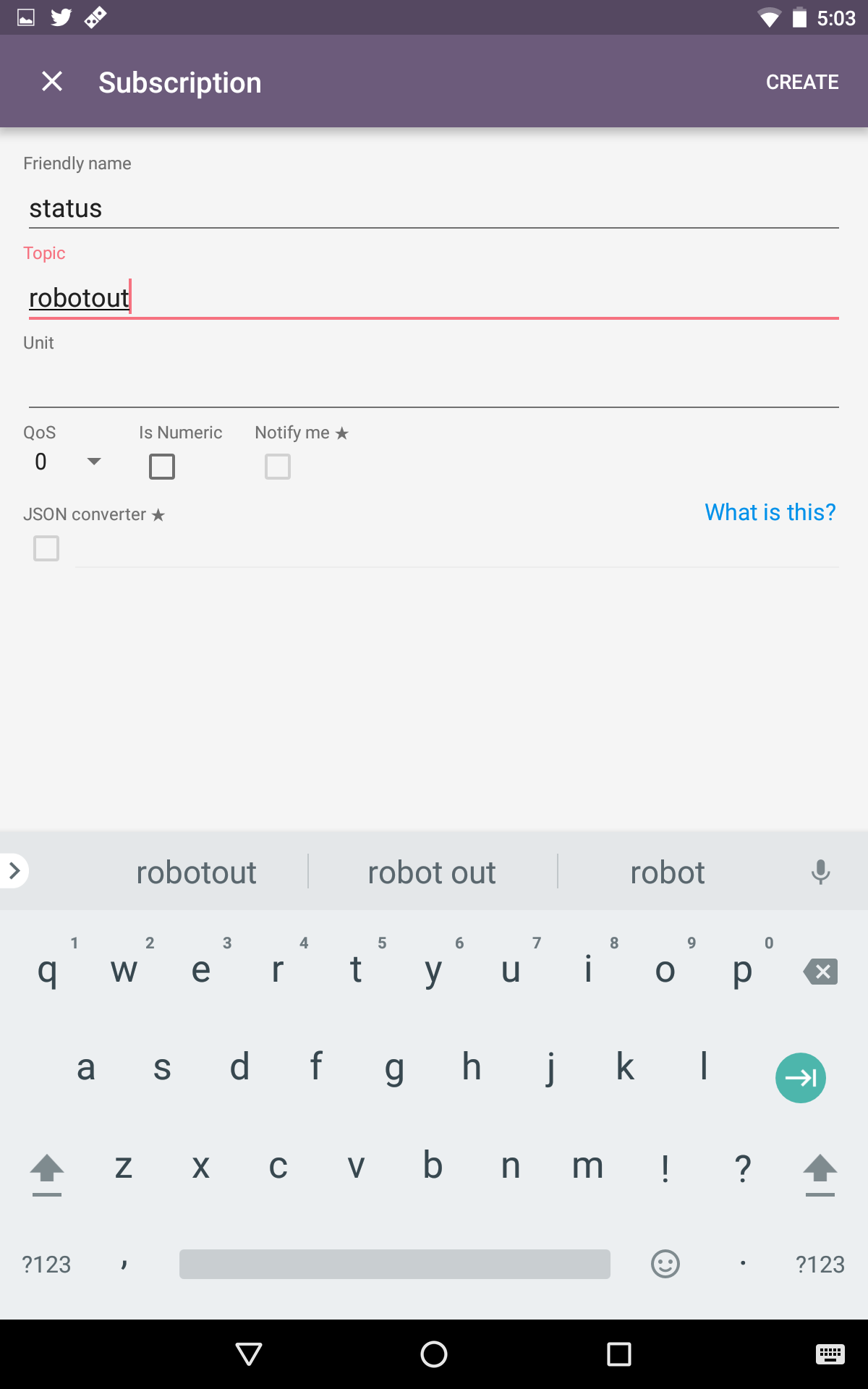
This app appears to do everthing I want, thou it doesn’t seem to let you move the buttons around. And the other limitation I see is the subscribe and publish are two different tabs.

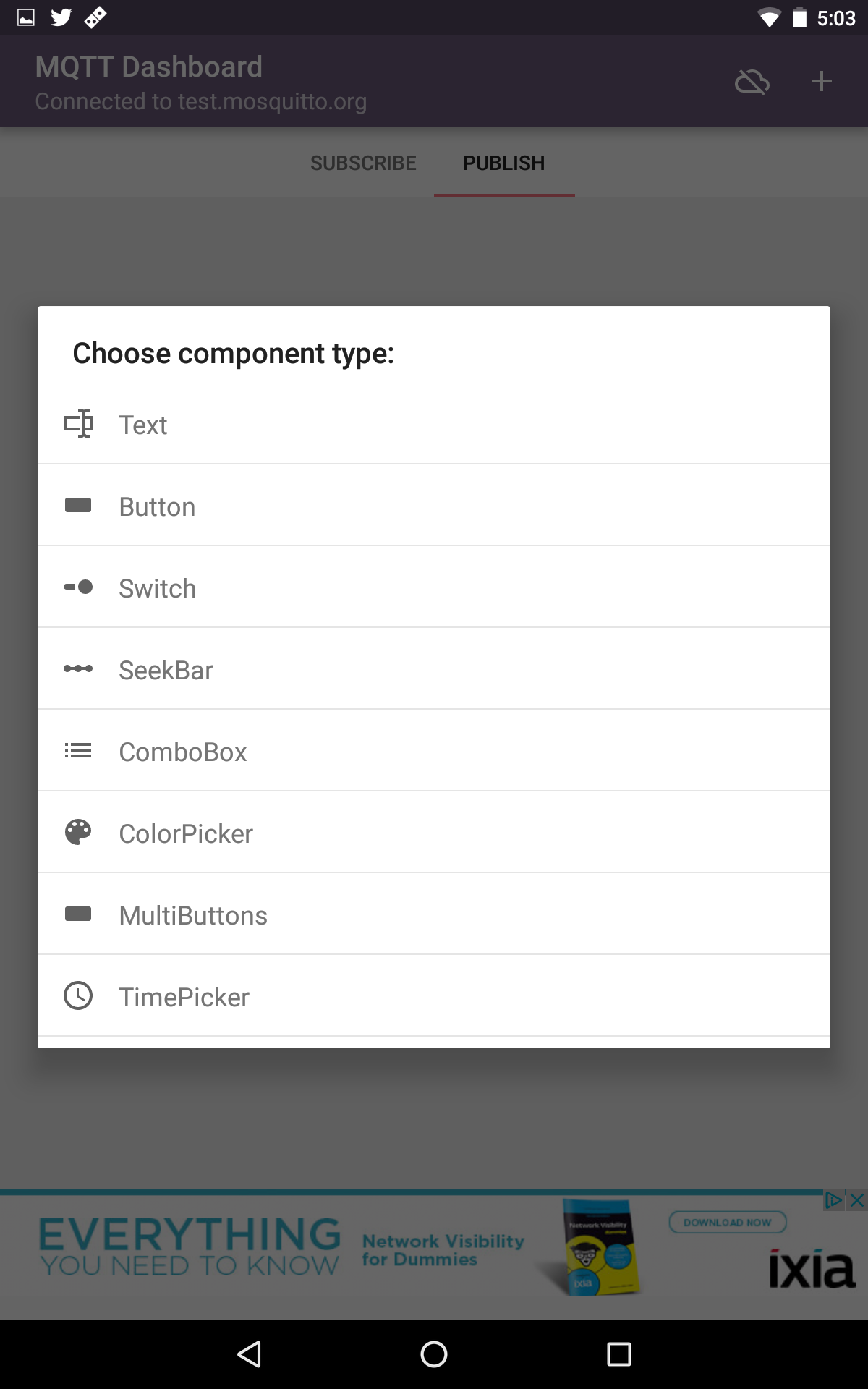
Still it’s easy to setup, lets you make buttons, switches, etc and lets you define what each button will publish. It lets you setup private or public MQTT brokers

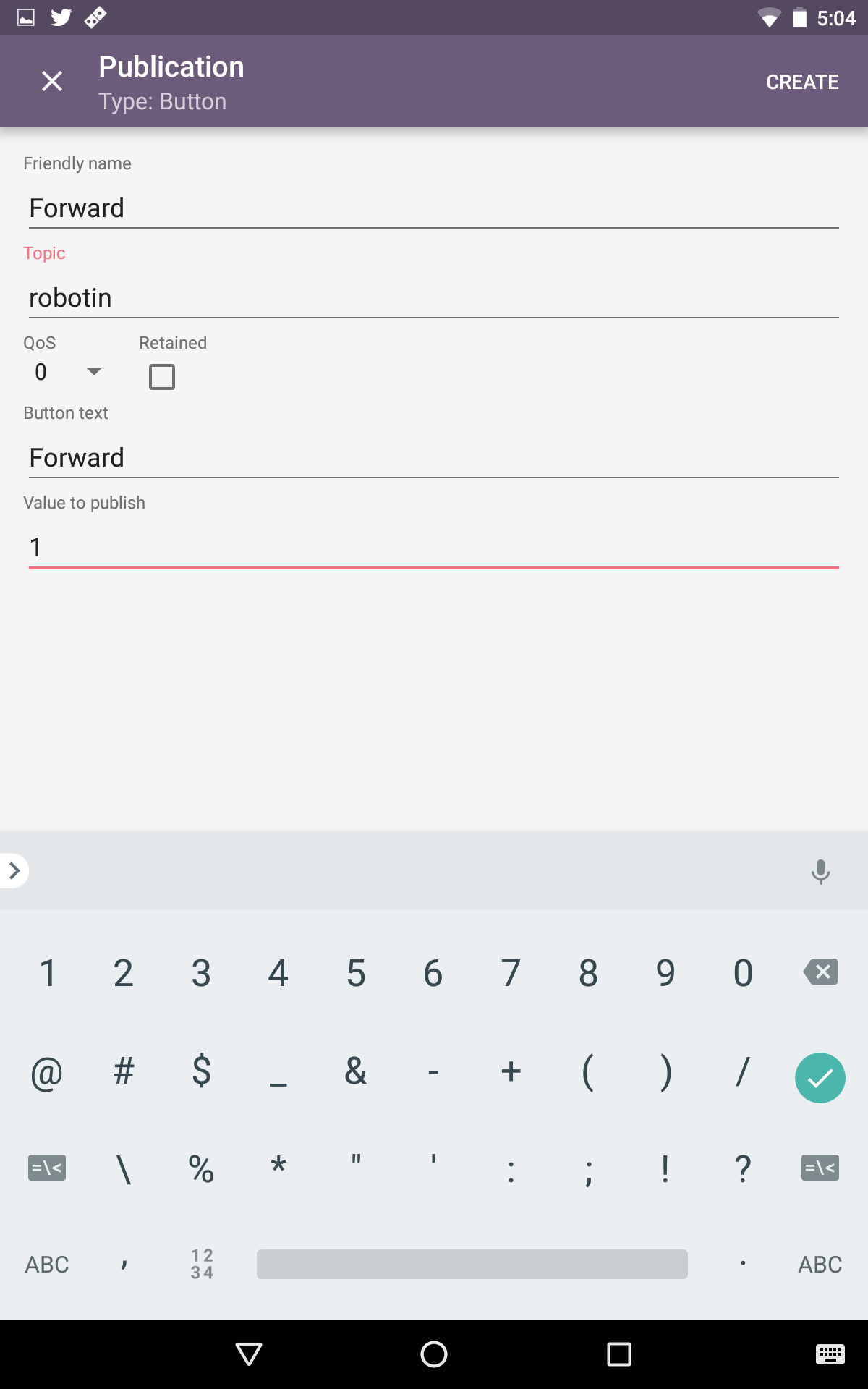
It is very easy to use, see the screen shots below.

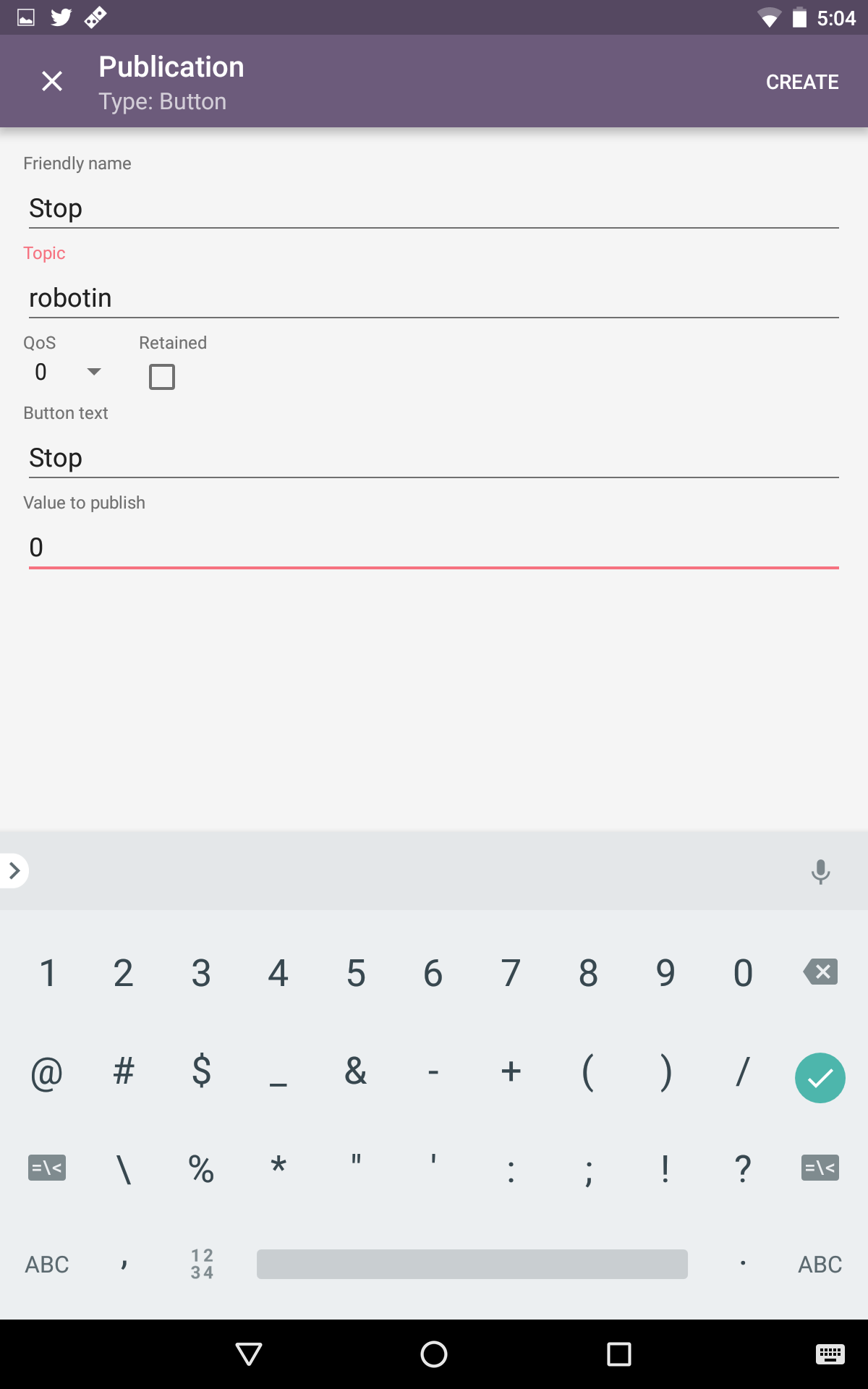


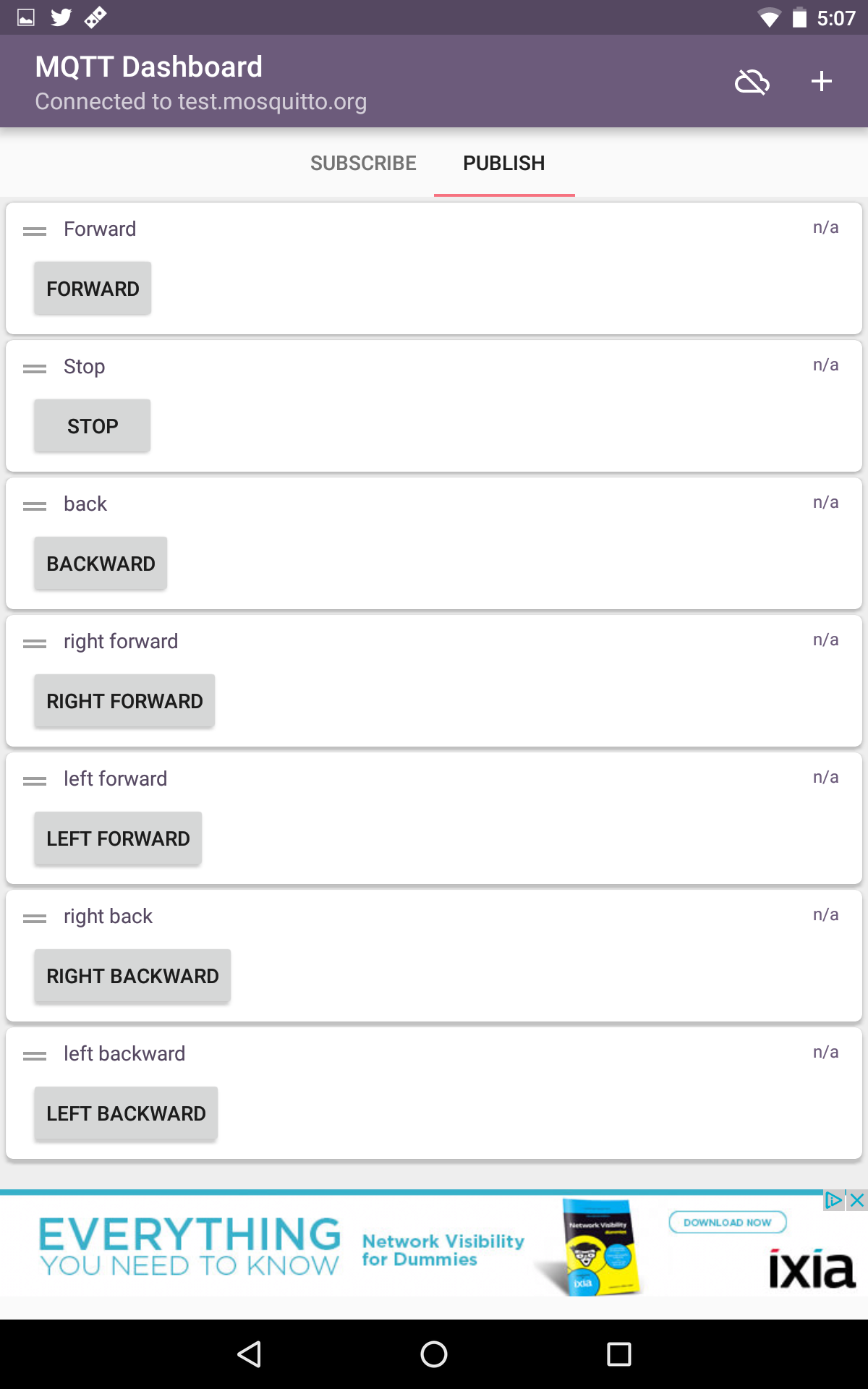


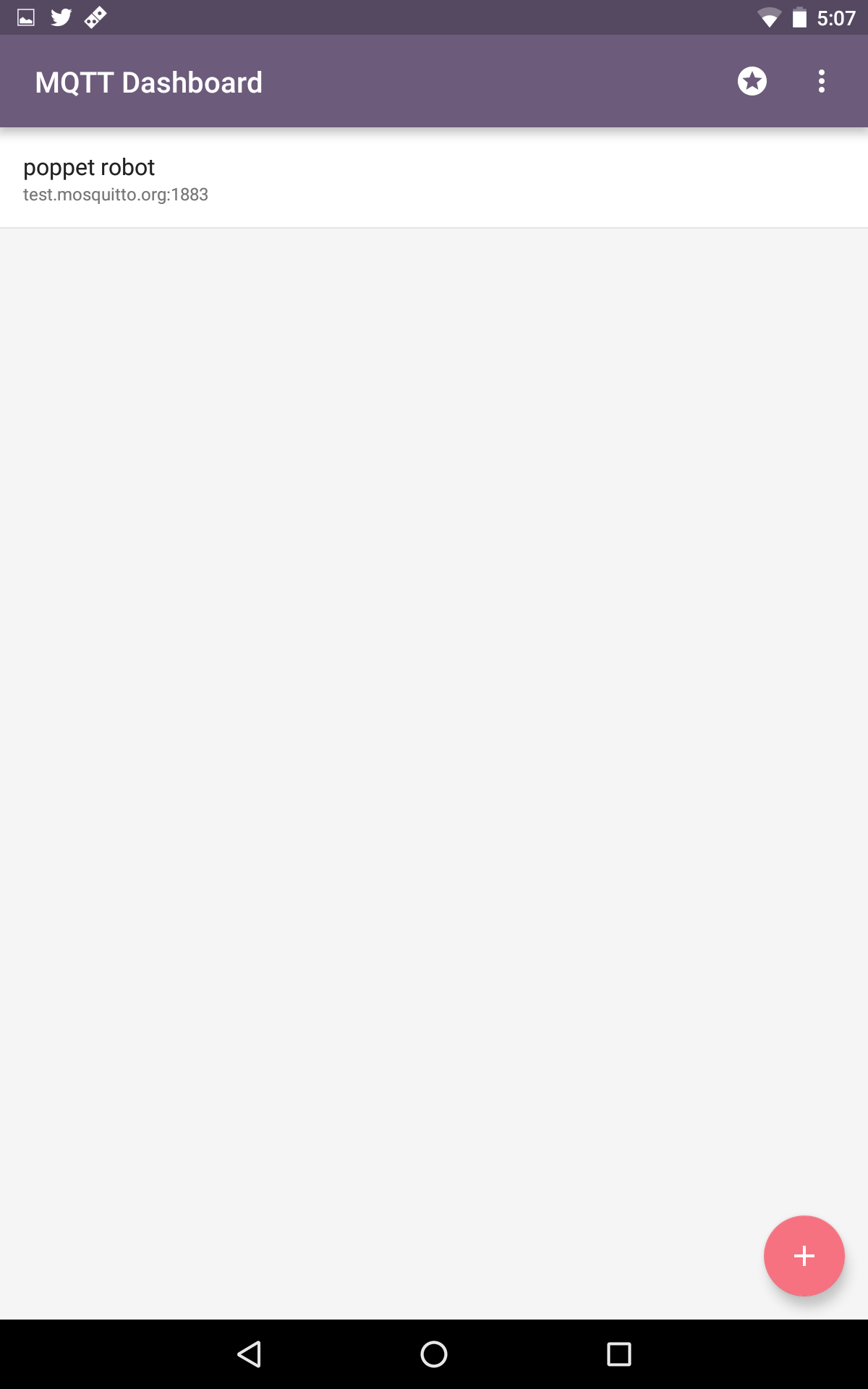












Some Mosquitto test broker information:

<http://test.mosquitto.org/>

More MQTT Arduino Libraries:

<https://github.com/knolleary/pubsubclient> (the 1st one and current one I use)

<https://github.com/256dpi/arduino-mqtt> (another client for arduino)

Crouton Dashboard - Crouton is very interesting, and something that I’d like to explore more. It makes buttons and text boxes based on what is sent from the client. It has a way to auto connect to various devices. I found it interesting while I was testing, but ultimately would have taken a pretty big rework of the sketch to work for this.

Thou, I do want to explore it more, this may not be the right project for it.

<http://crouton.mybluemix.net/crouton/gettingStarted>

<https://github.com/edfungus/Crouton>

Other things I’d like to try - Node RED for Interacting with Arduino.

<https://nodered.org/docs/hardware/arduino>