Commonly Arduino is used to work with Android over MQTT. But for an industrial setting Industruino is more practical. You don't need to use third party IoT platforms like Cayenne or Blynk and has the luxury to customize the solution according to your need.

Nevertheless, it was great fun to be able to control Industruino over MQTT from custom Android app, without using third party platforms.

Here are the specifics for Industruino side :

Industruino Environment Specification:
Board : Industruino D21G
Port : dev/tty/ACM0
Programmer: Atmel SAM-ICE

Serial Port Permission in ubuntu:

```
ls -l /dev/ttyACM0
sudo chmod a+rwx /dev/ttyACM0
```

==========
Following additional libraries are needed for Industruino, LCD display and MQTT to run basic sketch using the Ethernet Module:

Indio
UC1701
PubSubClient

Also make sure to have the adafruit Ethernet2 library. Note, the recommended Ethernet library doesn't work.

Need to configure for proper frequency in the following file:

3. In "Arduino/Libraries/Ethernet2/
src/utility/w5500.cpp", change
the SPI frequency to 4MHz.

Line 24: // SPI details
Line 25: "SPISettings wiznet_SPI_settings(4000000, MSBFIRST, SPI_MODE0);"
==========

Local MQTT broker:
After installing the mosquitto broker sample publishing and subscribing can be done in following way:

```
mosquitto_sub -h localhost -t industruinoOutTopic/#
mosquitto_pub -h localhost -t "weather/test" -m "hello"
```

To check topic status:

sudo netstat -tulpn | grep 1883

Third party broker:
I have found Industruino is working with HIVEMQ, CLOUDMQTT etc

Third party IoT platform like Cayenne:
Load cayenne library in the Arduino IDE and put your Cayenne credentials in the code. Also need to configure arduino IP address and Gateway based on your network. DNS and Subnet Mask should be unchanged. Also selected device is Industruino, not anything else. Then from Cayenne dashboard it was possible to see the readings.

==========

Basic Sketch to Publish and Subscribe to topics from Industruino. Thanks to (https://industruino.com/blog/our-news-1/post/industruino-mqtt-client-over-ethernet-35)

```
/*
  Basic MQTT example for Industruino D21G with Ethernet module
  based on the mqtt_basic example of the pubsubclient library

  This sketch demonstrates the basic capabilities of the library.
  It connects to an MQTT server as 'industruinoClient'
  - publishes "hello" to the topic "industruinoOutTopic"
  - publishes on_time to the topic "industruinoOutTopic" every 5
seconds
  - subscribes to the topic "industruinoInTopic", and switches on/off
LCD backlight on 1/0

  It will reconnect to the server if the connection is lost using a
blocking
```

```
  reconnect function. See the 'mqtt_reconnect_nonblocking' example for
how to
  achieve the same result without blocking the main loop.

*/

#include <SPI.h>
#include <Ethernet2.h>        // Industruino version of Ethernet2
#include <PubSubClient.h>

#include <UC1701.h>
static UC1701 lcd;

// Update these with values suitable for your network.
byte mac[] = {  0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xED };  // or use
Industruino unique MAC
IPAddress ip(192, 168, 0, 177);
IPAddress server(192, 168, 0, 17);
//char server[] = "m24.cloudmqtt.com";
//char user[] = "";
//char pass[] = "";

EthernetClient ethClient;
PubSubClient client(ethClient);
unsigned long timestamp;    // for regular publishing

void setup()
{
  SerialUSB.begin(115200);
  pinMode(26, OUTPUT);      // LCD backlight
  digitalWrite(26, HIGH);

  lcd.begin();
  lcd.print("Industruino MQTT demo");
  SerialUSB.println("Industruino MQTT demo");

  client.setServer(server, 1883);
  //client.setServer("test.mosquitto.org", 1883);   // public test
```

```
broker
  //client.setServer(server,15416);
  client.setCallback(callback);

  Ethernet.begin(mac, ip);
  SerialUSB.print("Ethernet started on IP: ");
  SerialUSB.println(Ethernet.localIP());
  lcd.setCursor(0, 1);
  lcd.print("thisIP:");
  lcd.print(Ethernet.localIP());
  lcd.setCursor(0, 2);
  lcd.print("server:");
  lcd.setCursor(0, 3);
  lcd.print(server);
  lcd.print(":1883");
}

void loop() {
  if (millis() - timestamp > 5000) {        // every 5 seconds do this
    if (!client.connected()) {
      reconnect();
    } else {
      String message;
      message = "on_time(ms):";
      message += String(millis());
      SerialUSB.println("SENDING [industruinoOutTopic]: " + message);
      lcd.setCursor(0, 5);
      lcd.print("SENT " + message + "      ");
      int len = message.length() + 1;
      char buf [len];
      message.toCharArray(buf, len);
      client.publish("industruinoOutTopic", buf);     // this function
does not take String as parameter
    }
    timestamp = millis();
  }
  client.loop();
}
```

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    SerialUSB.print("Attempting MQTT connection to server: ");
    SerialUSB.print(server);
    SerialUSB.print(" >>> ");
    lcd.setCursor(0, 5);
    lcd.print("connecting...      ");
    // Attempt to connect
    if (client.connect("industruinoClient")) {
      SerialUSB.println("connected");
      // Once connected, publish an announcement...
      client.publish("industruinoOutTopic", "hello from industruino");
      lcd.setCursor(0, 5);
      lcd.print("SENT hello      ");
      // ... and resubscribe
      client.subscribe("industruinoInTopic");
    } else {
      SerialUSB.print("failed, rc=");
      SerialUSB.print(client.state());
      SerialUSB.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void callback(char* topic, byte* payload, unsigned int length) {
  SerialUSB.print("RECEIVING [");
  SerialUSB.print(topic);
  SerialUSB.print("]: ");
  lcd.setCursor(0, 5);
  lcd.print("RECEIVED ");
  for (int i = 0; i < length; i++) {
    SerialUSB.print((char)payload[i]);
    lcd.print((char)payload[i]);
  }
  lcd.print("              ");
```

```
  SerialUSB.println();

  if ((char)payload[0] == '0') {
    digitalWrite(26, LOW);
  }
  if ((char)payload[0] == '1') {
    digitalWrite(26, HIGH);
  }
}
```

Hope to write about Android part in another post.