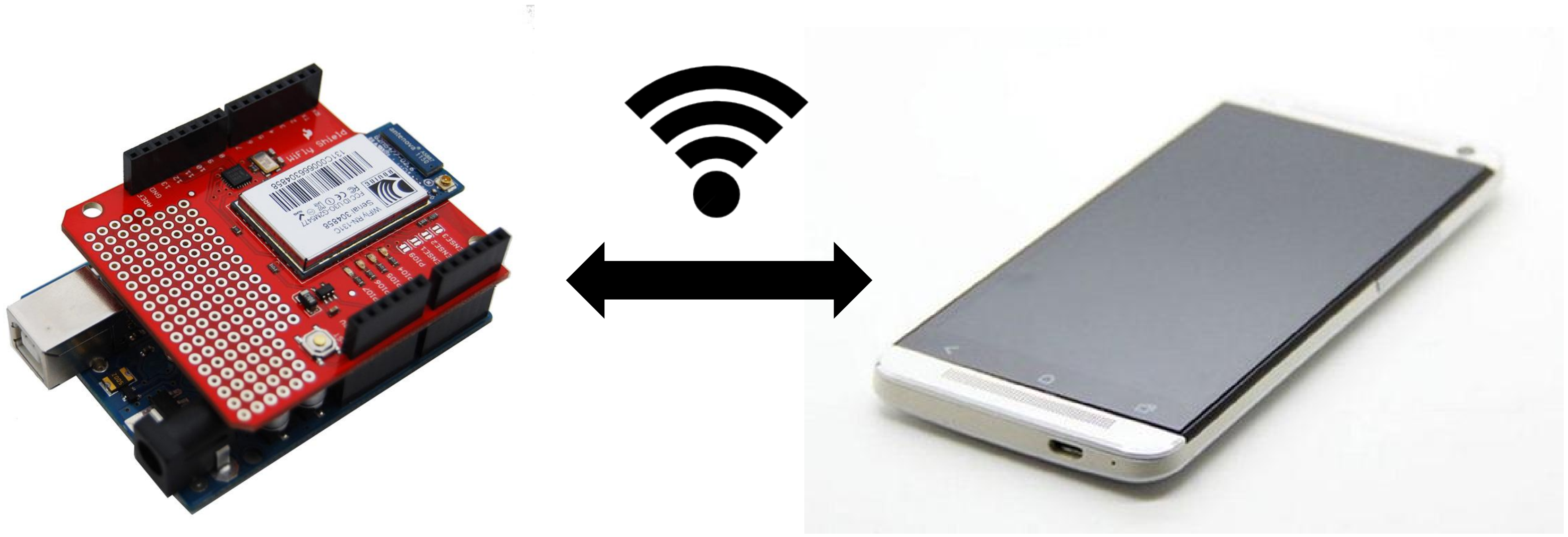
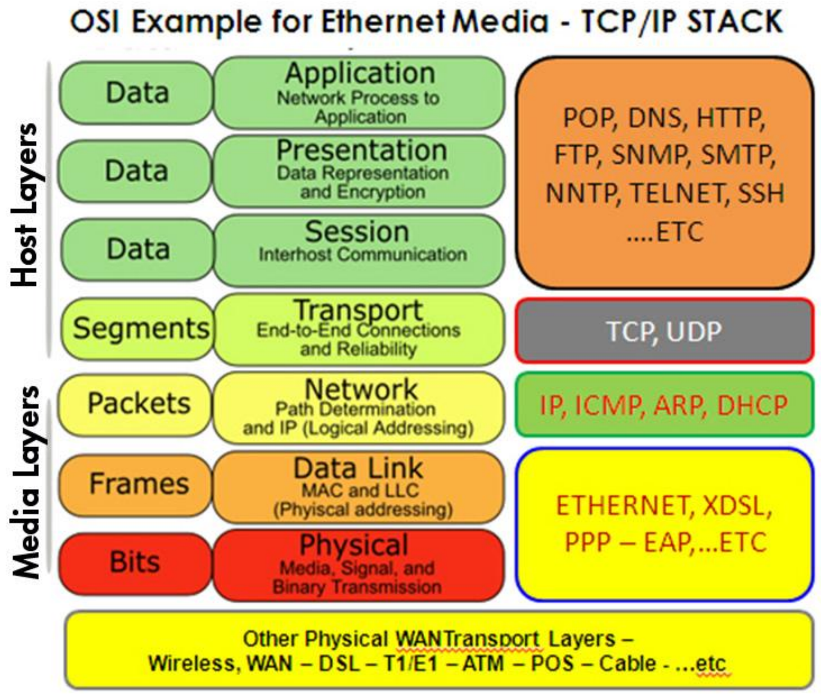
Arduino-Android Communication

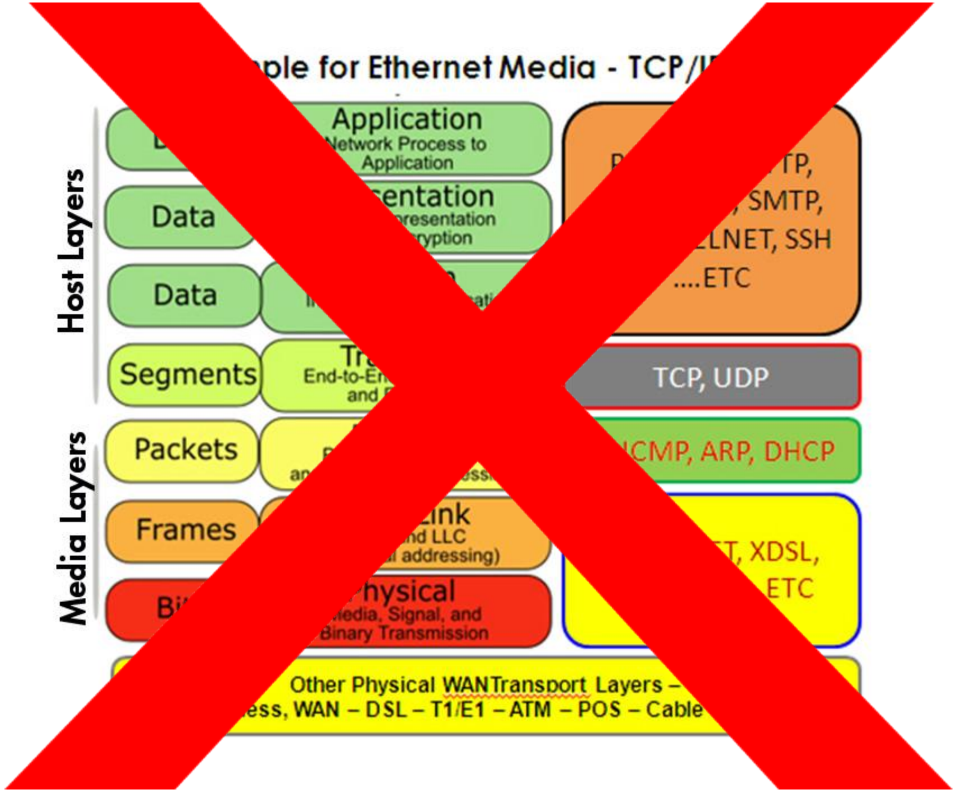
How to make WiFi *really* work.

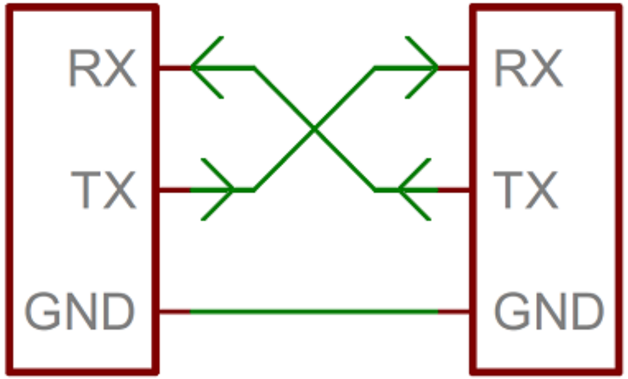


The goal - simplify.


OSI Example for Ethernet Media - TCP/IP STACK
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Phyiscal addressing)
Bits
Physical
Media, Signal, and Binary Transmission
POP, DNS, HTTP, FTP, SNMP, SMTP, NNTP, TELNET, SSH ....ETC
TCP, UDP
IP, ICMP, ARP, DHCP
ETHERNET, XDSL, PPP – EAP,...ETC
Other Physical WANTransport Layers – Wireless, WAN – DSL – T1/E1 – ATM – POS – Cable - ...etc

The goal - simplify.





The goal - simplify.

```
String msg = "ROLL_ANGLE 12";
send_message(msg);
```

```
got_message(String msg) {
   ...
}
```

The goal - simplify.

```
String msg = “ROLL_ANGLE 12”;
send_message(msg);
```

```
String msg = “EMERGENCY_STOP”;
send_message(msg);
```

```
got_message(String msg) {
  ...
}
```

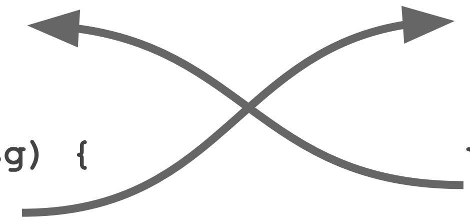
```
got_message(String msg) {
  ...
}
```

The starting point

```
void got_message(String msg) {
  // Do something!
}

void send_message(String msg) {
  connection.write(msg);
}

while(True) {
  String msg = connection.read();
  got_message(msg);
}
```

The starting point

```
void got_message(String msg) {
  // Do something!
}

void send_message(String msg) {
  connection.write(msg);
}

while(True) {
  String msg = connection.read();
  got_message(msg);
}
```



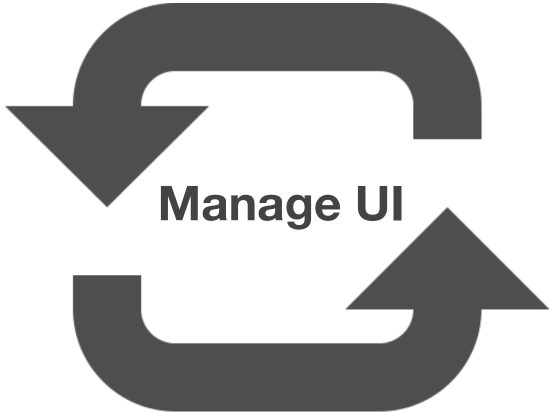
```
void got_message(String msg) {
  // Do something!
}

void send_message(String msg) {
  connection.write(msg);
}

while(True) {
  String msg = connection.read();
  got_message(msg);
}
```

Complication - reads take time.

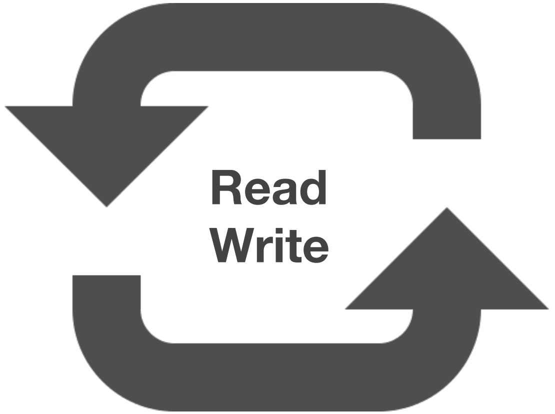
```
while(True) {

  String msg = connection.read();
  got_message(msg);

  // This will not run fast
  other_stuff();
}
```

Complication - reads take time.

Solution - read in a separate thread.



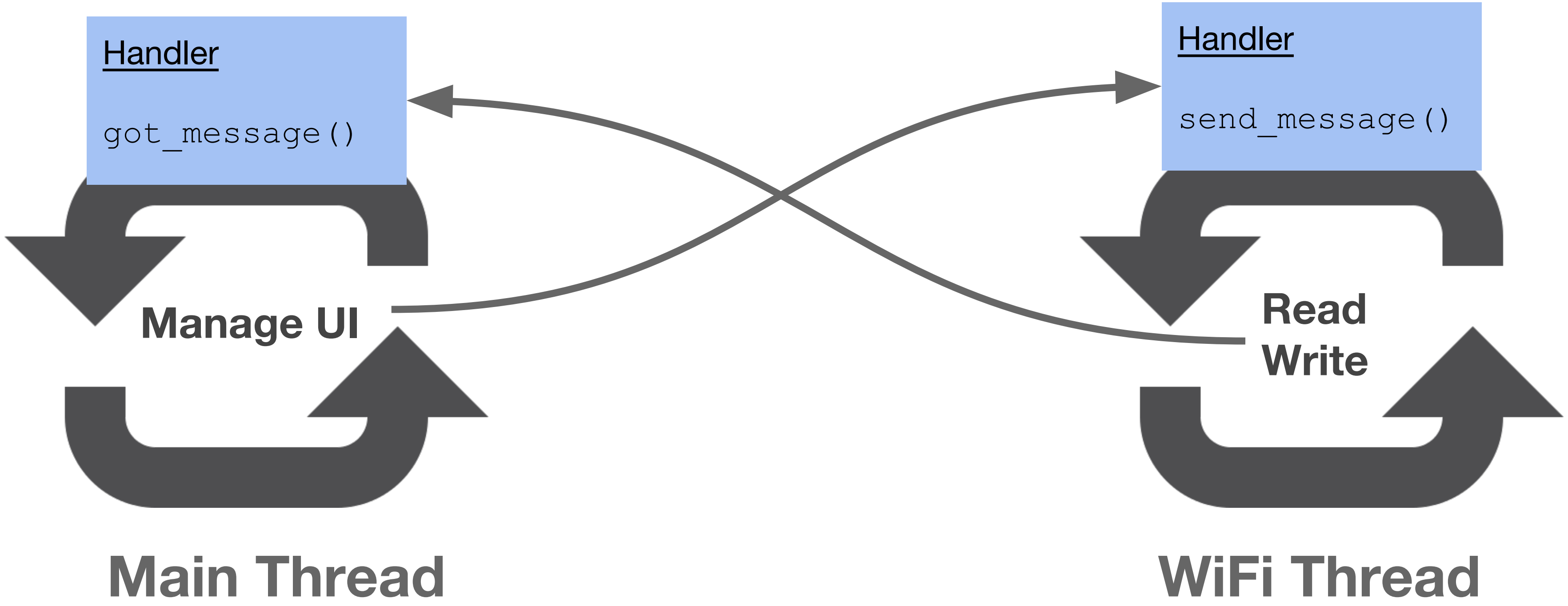
Main Thread



WiFi Thread

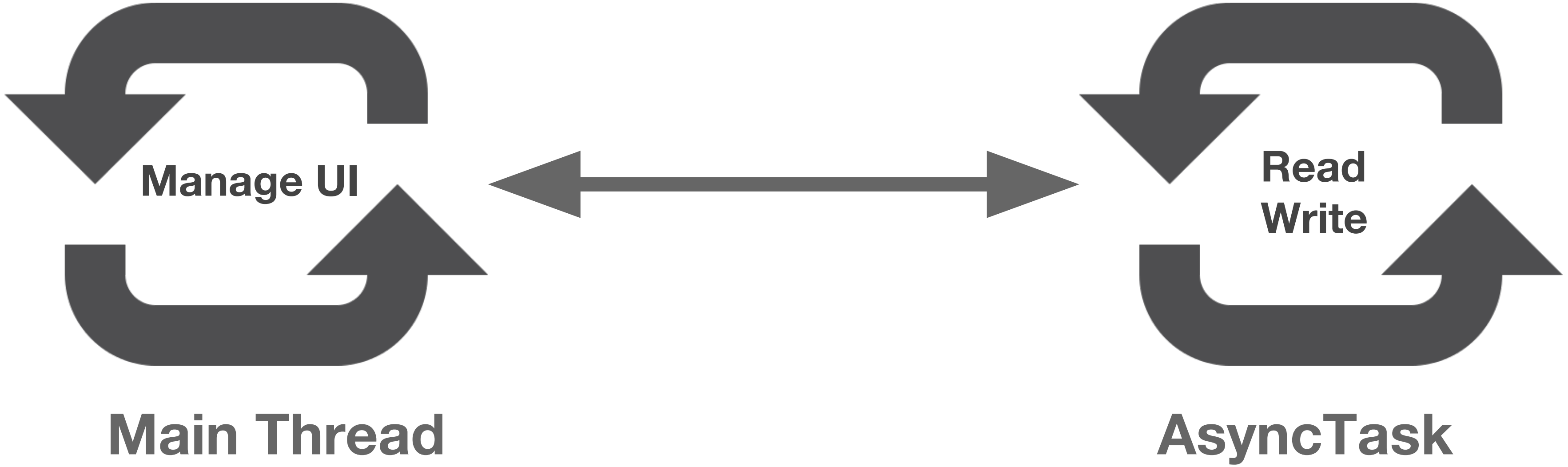
Complication - reads take time.

Solution - read in a separate thread.

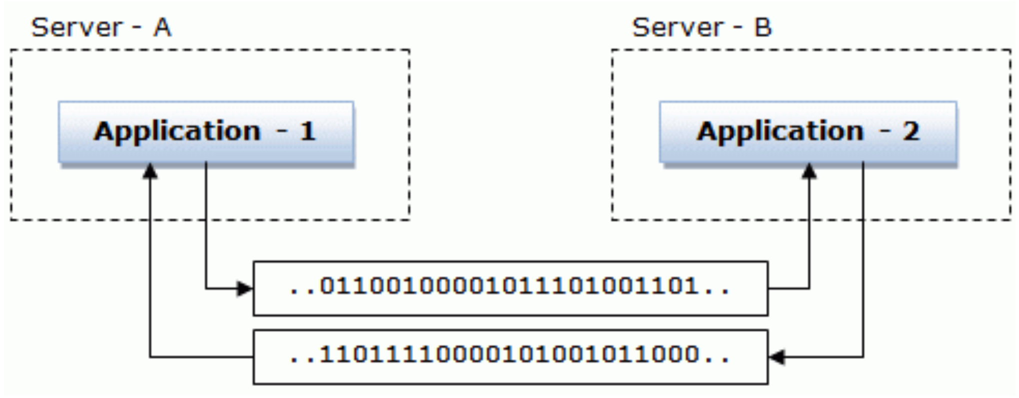


Complication - reads take time.

Solution - read in a separate thread.



Complication - streams are continuous.



```
send_message("Lollapalooza");
```

```
got_message("Loll");
got_message("apalooz");
got_message("a");
```

Complication - streams are continuous.

Solution - use a message delimiter.

```
send_message(String msg) {
  connection.write(msg);
  connection.write(‘\n’);
}
```

Complication - streams are continuous.

Solution - use a message delimiter.

```
send_message(String msg) {
  connection.write(msg);
  connection.write(‘\n’);
}
```

→

```
String rx_buffer = "";

while(connection.available()) {

  char c = connection.read();
  if(c == '\n') {
    got_message(rx_buffer);
    rx_buffer = "";
  } else {
    rx_buffer += c;
  }
}
```

Complication - streams are continuous.

Solution - use a message delimiter.

```
send_message(String msg) {
  connection.write(msg);
  connection.write(‘\n’);
}
```

$\longrightarrow$

```
String rx_buffer = "";

while(connection.available()) {

  char c = connection.read();
  if(c == '\n') {
    got_message(rx_buffer);
    rx_buffer = "";
  } else {
    rx_buffer += c;
  }
}
```

```
send_message(“Lollapalooza”);
```

$\xrightarrow[\texttt{“Lollapalooza\textbackslash n”}]{}$

```
got_message(“Lollapalooza”);
```

Complication - detecting disconnections.

Complication - detecting disconnections.

Solution - use a ping/timeout system.

```
int now = time();
if (now - last > PING_TIME) {
  send_message(PING_MSG);
  last = now;
}
```

Complication - detecting disconnections.

Solution - use a ping/timeout system.

```
int now = time();
if (now - last > PING_TIME) {
  send_message(PING_MSG);
  last = now;
}
```

$\longrightarrow$

```
if(msg == PING_MSG) {
  last = time();
}
```

Complication - detecting disconnections.

Solution - use a ping/timeout system.

```
int now = time();
if (now - last > PING_TIME) {
  send_message(PING_MSG);
  last = now;
}
```

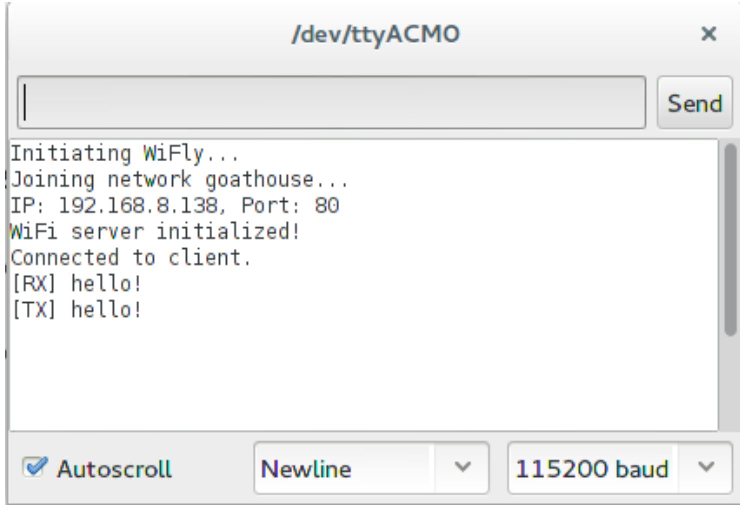
```
if(msg == PING_MSG) {
  last = time();
}
```

<u>**Check:**</u>

```
int now = time();
if(now - last > TIMEOUT) {
  disconnected();
}
```

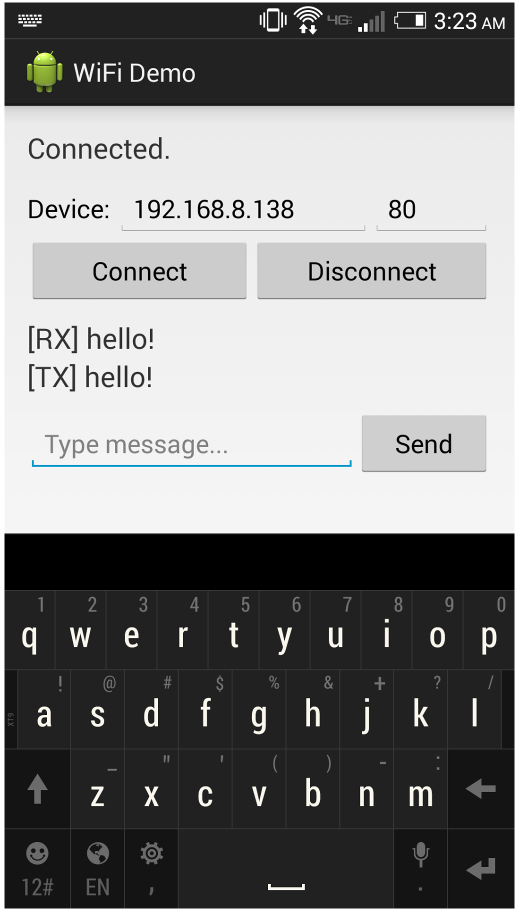
Demo Application

https://github.com/hmartiro/android-arduino-wifi/



```
Initiating WiFly...
Joining network goathouse...
IP: 192.168.8.138, Port: 80
WiFi server initialized!
Connected to client.
[RX] hello!
[TX] hello!
```

Autoscroll: ✓ — Newline — 115200 baud



Connected.

Device: 192.168.8.138 80

[RX] hello!
[TX] hello!

In the wild - for your consideration.

+ Bandwidth limits
+ Sending binary data
+ Parsing arguments in messages
+ Communicating with web APIs

Arduino-Android Communication

The end.