WiFröst: Bridging the Information Gap for Debugging of Networked Embedded Systems

Will McGrath, Jeremy Warner, Mitchell Karchemsky, Andrew Head, Daniel Drew, and Bjoern Hartmann
IoT Device Debugging

- Defective part?
- Peripheral not configured?
- Protocol error?
- Wiring issue?
- Program logic bug?
IoT Device Debugging

- Microcontroller
- GPIO+Buses
- Circuit
IoT System Debugging

Network
TCP/IP
Microcontroller
GPIO+Buses
Circuit
IoT System Debugging

- Server Down or Overloaded?
- Poorly Formatted API Request?
- Bad Server Address?
- Router Off or Overloaded?
- Intermittent Radio Interference?
- Out of Range?
- Bad Network Configuration or Credentials?
- Wrong or Invalid API Key?
- Protocol error?
- Wiring issue?
- Peripheral not configured?
- Defective part?
- Program logic bug?
IoT System Security

- Using Deprecated API?
- API Key / Password Sent in Plaintext
- Insecure Protocol Usage?
- Outdated firmware?
- Abnormal Network Traffic?
- Protocol error?
- Wiring issue?
- Peripheral not configured?
- Program logic bug?
- Defective part?
Existing Approaches

- JTAG debugger
- Logic analyzer
- Network analyzer
Each tool observes only a small slice of device state, and it is generally difficult to trace event causality across tool boundaries due to different representations and a lack of time synchronization.
Ubiquitous instrumentation, visualization, and checking can help users more effectively debug and understand IoT systems across hardware, software, and device boundaries.
Instrumentation

- Router and server Instrumentation
- Program and library instrumentation
- WiFröst

Network
- TCP/IP
- Microcontroller

- GPIO+Buses
- Circuit
Our Approach

Device

Router (MITM proxy)

Server

Network analyzer

Logic analyzer

Capture PC

WiFröst IDE

DB
Embedded Systems and Electronics Debugging

• Visible Breadboard [Ochiai, 2014]
• CurrentViz [Wu et al., 2017]
• Scanalog [Strasnick et al., 2017]

Embedded System Resource Usage

• NesC [Gay et al., 2014]
• Quanto [Fonseca et al., 2008]
Network and Distributed System Debugging

- ndb [Handigol et al., 2012]
- Netsight [Handigol et al., 2014]
- Whodunit [Chanda et al., 2007]
- xtrace [Fonseca et al., 2007]
- OFRewind [Wundsam et al., 2011]
- Pinpoint [Chen et al. 2002]
- Dapper [Sigelman et al. 2010]
- Pip [Reynolds et al., 2006]
Integrated Debugging Tools

Toastboard, 2016

Bifröst, 2017

WiFröst, 2018
Coverage

“Invalid parameter to web API call”

“Component is backwards”

“Reading from a pin set to output”
WiFrööst IDE

User Code

Network Connections

Event Information

User Code

Function Stack

Application Health Bars

Check Results

Console
WiFröst IDE

User Code

Function Stack

Event Information

Console

Network Connections

Check Results

Application Health Bars
Something seems amiss. Look at the Issues pane to see what's going wrong and tips on how to fix it.

156

Making HTTP request
Something seems amiss. Look at the Issues pane to see what's going wrong and tips on how to fix it.
Visual Debugging

ERROR

OK (200)
Health Bars
Introduction
Related Projects
IDE Features
Checks
Walkthrough
Implementation
Future Work
## Checks

<table>
<thead>
<tr>
<th>Issue</th>
<th>Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device out of memory</td>
<td>![ ]</td>
</tr>
<tr>
<td>Incorrect Wi-Fi SSID</td>
<td>![ ]</td>
</tr>
<tr>
<td>Incorrect Wi-Fi password</td>
<td>![ ]</td>
</tr>
<tr>
<td>Bad host name</td>
<td>![ ]</td>
</tr>
<tr>
<td>Weak connection to router</td>
<td>![ ]</td>
</tr>
<tr>
<td>Buggy web API usage</td>
<td></td>
</tr>
<tr>
<td>Can't access Internet</td>
<td></td>
</tr>
<tr>
<td>Incorrect authentication</td>
<td></td>
</tr>
</tbody>
</table>
Generic Checks

Issues

<table>
<thead>
<tr>
<th>Time</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>Error in API call: { &quot;type&quot;: 2, &quot;address&quot;: &quot;/lights/4/state/&quot;, &quot;description&quot;: &quot;body contains invalid json&quot; }</td>
</tr>
</tbody>
</table>

Fix
- Check your API call on line 156.
Custom Checks

```javascript
{
  name: "bad-host",
  priority: 1,
  check: function(events) {
    let locations = []
    for (let i = 0; i < events.length; i++) {
      let ev = events[i]
      if (ev.type === "deviceIns" && ev.value === "DNSResolve" && ev.name === "0") {
        locations.push(ev)
      }
    }
    return locations
  },
  describeError: function(event, line) {
    return "The host you're sending a request to couldn't be found."
  },
  describeFix: function(event, line) {
    return "Look at the host that you're trying to query on line " + 
    line.number + ". The host name may be wrong."
  },
  getFlagLocations: function(event, line) {
    return { error: ["line"] }
  }
}
```
Introduction
Related Projects
IDE Features
Checks
Walkthrough
Implementation
Future Work
Walkthrough - Weather
Walkthrough – Connecting to WiFi

```c
#include <ArduinoJson.h>
#include <WiFi.h>

const size_t bufferSize = JSON_OBJECT_SIZE(0) +
                         JSON_OBJECT_SIZE(1) + JSON_OBJECT_SIZE(2) + JSON_OBJECT_SIZE(3) +
                         JSON_OBJECT_SIZE(4) + JSON_OBJECT_SIZE(5) + JSON_OBJECT_SIZE(6) +
                         JSON_OBJECT_SIZE(7) + JSON_OBJECT_SIZE(8) + JSON_OBJECT_SIZE(9) +
                         JSON_OBJECT_SIZE(10) + 2240;

const char APIKEY[] = "bxw1lozodpzwq3qyvq";
char ssid[] = "Net2";
char passwd[] = "password";
int status = WiFi_IDLE_STATUS;
WIFIClient client;
unsigned long lastConnectionTime = 0; // last time you connected to the server, in milliseconds
IPAddress host[2] = {192.168.2.123, 255.255.255.0};
const char base[] = "https://example.com";
String data = ""; // String with json data
String to = "No data";
const unsigned long postingInterval = 10L * 1000L; // delay between updates, in milliseconds

void setup()
{
  WiFi.setPins(8, 7, 4);
  Serial.begin(115200);
  Serial.println("Serial initialized");
  if (WiFi.status() == WIFI_NOT_IN_RANGE) {
    Serial.println("No network found");
    return;
  }

  WiFi.begin(ssid, passwd);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("Connecting to ");
    Serial.println(ssid);
  }

  Serial.println("Connected to ");
  Serial.print("IP Address:");
  Serial.println(WiFi.localIP());
}

void loop()
{
  // Log Serial initialized
  if (Serial.available() > 0) {
    Serial.print("Log: ");
    Serial.print(Serial.readString());
    Serial.println(" initialized");
  }

  // Log Attempting to connect to open SSID
  Serial.println("Log: Attempting to connect to open SSID");

  // Check connection status
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Connected to ");
    Serial.print("IP Address:");
    Serial.println(WiFi.localIP());
    Serial.println(" Attempting to connect to open SSID");
  } else {
    Serial.println("No network found");
  }

  // Check connection status periodically
  if (millis() - lastConnectionTime > postingInterval) {
    lastConnectionTime = millis();
    Serial.println("Log: Attempting to connect to open SSID");
  }
}```
Walkthrough – Connecting to WiFi

Log: Attempting to connect to open SSID:
   Log: Net2

Log: Attempting to connect to open SSID:
   Log: Net2

Log: Attempting to connect to open SSID:
   Log: Net2

Issues

<table>
<thead>
<tr>
<th>Time</th>
<th>Error</th>
<th>Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>Invalid Wi-Fi password.</td>
<td>Double-check the password you're using to connect to the router.</td>
</tr>
</tbody>
</table>
Walkthrough – Incorrect Domain Name

```c
#include <arduinoJson.h>
#include <SPI.h>

const size_t bufferSize = JSON_OBJECT_SIZE(8) + JSON_OBJECT_SIZE(1) + JSON_OBJECT_SIZE(2) + 2*JSON_OBJECT_SIZE(1) + JSON_OBJECT_SIZE(8) + JSON_OBJECT_SIZE(56) + 2400;

const char APIKEY[] = "bxw+lozodpq6vq";
char ssid[] = "Net2";
int status = WL_IDLE_STATUS;
WiFiClient client;
unsigned long lastConnectionTime = 0; // last time you connected to the server, in milliseconds
//IPAddress host(192,168,1,123);
const char host[] = "api.wunderground.com";
const char host[] = "thisdoesnotexist.com";
String data = ""; // String with Json data
String fo = "No data";
const unsigned long postingInterval = 10L * 1000L; // delay between updates, in milliseconds
char buffer[4000];

void setup(){
  WiFi.setScan(true, 4);
  Serial.begin(115200);
  Serial.println("Serial initialized");
}

void loop(){
  if (WiFi.status() == WL_CONNECTED) {
    // Do something here...
    // Send data...
  } else {
    // No connection...
  }
}
```

**Console Logs**

// Serial output: to explore, click on timestamps or use arrow keys to explore.

```
0.008s */
0.008s */ (Beginning of trace) */
0.043s */ Serial initialized
0.043s */ Attempting to connect to open SSID:
0.043s */ Net2
0.043s */ Attempting to connect to open SSID:
0.043s */ Net2
10.836s */ You're connected to the network
16.936s */ connecting to
16.936s */ api.wunderground.com
30.944s */ connection failed
31.094s */ connecting to
31.094s */ api.wunderground.com
```

**Issues**

The host you're sending a request to couldn't be found. Look at the host that you're trying to query on line 43. The host name may be wrong.
Walkthrough – Incorrect Domain Name

```cpp
#include <arduinojson.h>
#include <SPI.h>

const size_t bufferSize = JSON_OBJECT_SIZE(0) +
  JSON_OBJECT_SIZE(1) + JSON_OBJECT_SIZE(2) + JSON_OBJECT_SIZE(3) + JSON_OBJECT_SIZE(4) +
  JSON_OBJECT_SIZE(56) + 2240;

css[0] = "Net2";
char ssid[] = "Net2\0\0";

int status = WL_IDLE_STATUS;
WiFiClient client;
unsigned long lastConnectionTime = 0;
// last time you connected to the server, in milliseconds
IPAddress host(192,168,1,22);
const char host[] = "api.wuunderground.com";
const char host[] = "thisisnotexist.com";

String data = ""; // String with json data
String fo = "No data";

const unsigned long postingInterval = 60L * 1000L; // delay between updates, in milliseconds

char buffer[4000];

void setup(){
  WiFi.configIP(8,7,4);
  Serial.begin(115200);
  Serial.println("Serial initialized");
}

void loop(){
  // Send request to server
  http.begin();
  http.addHeader("Content-Type", "application/json");
  http.end();

  if (http.status() == HTTP_OK) {
    // Parse JSON
    // Print the JSON
  } else {
    // Error
  }
}
```

**Console Logs**

```
// Serial output: to explore, click on timestamps
// or use arrow keys to explore.

/* Time */
/* 0.0008 */ // (Beginning of trace )
/* 0.0008 */ // Serial initialized
/* 0.543x */ // Attempting to connect to open SSID:
/* 0.543x */ // Net2
/* 5.675x */ // Net2
/* 5.676 */ // You’re connected to the network
/* 10.134x */ // connecting to
/* 10.135x */ // api.wuunderground.com
/* 36.094 */ // connection failed
/* 31.094 */ // api.wuunderground.com
```

**Issues**

- The host you’re sending a request to couldn’t be found.
- Look at the host that you’re trying to query on line 60. The host name may be wrong.
Walkthrough – Incorrect Domain Name

1. Something seems amiss. Look at the Issues pane to see what’s going wrong and tips on how to fix it.

Code Editor

```
#include <arduinojson.h>
#include <ESP8266WiFi.h>

const size_t bufferSize = JSON_OBJECT_SIZE(8) +
  JSON_OBJECT_SIZE(5) + JSON_OBJECT_SIZE(10) +
  JSON_OBJECT_SIZE(5) + JSON_OBJECT_SIZE(8) +
  JSON_OBJECT_SIZE(56) + 248;

char APIKEY[] = "bxwr-lzodpq6qvg";
char ssid[] = "Net2";
char pass[] = "Net212345678";
int status = WL_IDLE_STATUS;
WiFiClient client;
unsigned long lastConnectionTime = 0; // last time you connected to the server, in milliseconds
IPAddress host(192, 168, 2, 123);
const char host[] = "api.wunderground.com";
String data = ""; // String with json data
String fo = "no data";
const unsigned long postingInterval = 10L * 1000L; // delay between updates, in milliseconds

void setup()
{
  delayTillNextUpdate();
  delayTillNextUpdate();
  WiFi.mode(WIFI_STA);
  WiFi.setFailSafe(8, 4);
  Serial.begin(115200);
  Serial.println("Serial initialized");
  if (WiFi.status() == WL_NO_SHUTDOWN)
```

Issues

- Time: 43
- Error: 5
- Fix: Look at the host that you’re trying to query on line 60. The host name may be wrong.

Console Logs

```
// Serial output: to explore, click on timestamps
// or use arrow keys to explore.

*/ 0.0008s */ (Beginning of trace ) */
*/ 0.0008s */ Serial initialized
*/ 0.0438s */ Attempting to connect to open SSID:
*/ 0.0438s */ Net2
*/ 0.5774s */ Attempting to connect to open SSID:
*/ 0.5774s */ Net2
*/ 5.6723s */ You’re connected to the network
*/ 10.3861s */ connecting to
*/ 16.3861s */ api.wunderground.com
*/ 36.3846s */ connection failed
*/ 31.0964s */ connecting to
*/ 31.0964s */ api.wunderground.com
```
Walkthrough – API Key

```java
client.readStringUntil("\n");
}
void httpRequest()
{
    client.flush();
    client.stop();
    delay(1000);
    Serial.println("connecting to ");
    Serial.println(host);
    if (client.connect(host, 80)) {
        Serial.println("Sending GET Request: ");
        client.print("GET /api/browilizedxactosql/conditions?q
        /CA/Berkely, Json HTTP/1.1\r\n\n");
        client.print("Host: api.wunderground.com \r\n\n");
        client.print("Connection: close\r\n\n");
    } else {
        Serial.println("connection failed");
    }
    delay(100);  
    void parseJSON(String json[])
    {
        Serial.println("Parsing JSON");
        DynamicJsonBuffer jsonBuffer(bufferSize);
        JsonObject root = jsonBuffer.parseObject(json);
        if (!root.success())
        {
```
Walkthrough – API Usage

```java
    } else {
        // read and discard everything else
        client.readStringUntil('n');
    }
```
Walkthrough – Correct Behavior

```cpp
void httpRequest()
{
    // read and discard everything else
    client.readStringUntil("\n");
}

// Serial output: to explore, click on timestamps
// or use arrow keys to explore.
```

```cpp
device
router
server
```

```cpp
Log: Serial initialized
```

```cpp
Log: Attempting to connect to open SSID:
```

```cpp
Log: Net2
```

```cpp
Log: Attempting to connect to open SSID:
```

```cpp
Log: Net2
```

```cpp
Log: Attempting to connect to open SSID:
```

```cpp
Log: Net2
```

WiFröst - Implementation

Device

Logic Analyzer

Router (Mitmproxy)

Capture PC

Server

WiFröst IDE

DB
Compilation

User Code

Build

Function Lines

Inst.

Line #, Core Libs, RAM Use

Run

Database

Logic Analyzer

Routing

WiFröst IDE

Compilation

Build

Run
Network Instrumentation

Device

Connection Begin
GET / HTTP/1.1
Content-Type: application/json
Accept: */*
x-wftag: 0123

Connection End

Router (Mitmproxy)

Request Received

Response Received
HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
Connection: close
Expires: Mon, 1 Aug 2011 09:00:00 GMT
x-wftag: 0123

Request Handled (if instrumented)

Server

GET / HTTP/1.1
Content-Type: application/json
Accept: */*

x-wftag: 0123

HTTP/1.1 200 OK
Server: nginx
Content-Type: application/json
Connection: close
Expires: Mon, 1 Aug 2011 09:00:00 GMT
x-wftag: 0123
Time Domains

Device

Router (Mitmproxy)

Server

Logic Analyzer

Capture PC

WiFröst IDE

DB

μs resolution

ms resolution
Time Scaling

Actual

Logical
Automatic code and network instrumentation, checking, and trace visualization can help users better understand and debug their networked embedded devices.
Introduction
Related Projects
IDE Features
Checks
Walkthrough
Implementation
Future Work
Future Work

Router and server instrumentation
Program and library instrumentation
Microcontroller pin capture
Instrumented Breadboard

Network
TCP/IP
Microcontroller
GPIO+Buses
Circuit
Questions?
extra slides
buried/reference
void setAllLights(uint8_t hue, uint8_t sat){
    for(uint i = 0; i < 40; i++){
        setColor(i, hue, sat);
    }
}

void alarmLights(){
    int alarmDelay = 1000;
    setAllLights(0, 200);
    delay(alarmDelay);
    setAllLights(255, 200);
    delay(alarmDelay);
}

boolean checkSecurity(){
    if(digitalRead(securityWire) == 1) return true;
    return false;
}

void checkAlarm(){
    while (client.connected()) {
        String data = client.readStringUntil('"');
        Serial.println(gettingData);
    }
}

void loop() {
More related work…

• Bjoern said that he would like 2-3 slides of related work based on a desire for tying more projects together in a cohesive manner.
Preliminary User Feedback

Interfacing with the Phillips Hue API from Arduino using WiFröst

Users made effective use of the visualization and automatic checks for guidance

“Auto-formatting logs is nice, it’s work done for me.”

“Helpful for not only debugging, but also learning”

“Takes away the headache of finding [information] manually”
Our Integrated Electronic Debugging Tools

Instrumentation
- Instrumented Breadboard
- Microcontroller pin capture
- Program and library instrumentation
- Router and server instrumentation

Checking
- Ne
- TC
- Microc
- GPIO
- Ci

Router and server Instrumentation

Instrumentation

Wifröst

Bifröst

Toastboard

Wifröst
WiFröst: Exploration

User Code

Function Stack

Event Information

Console

Network Connections

Check Results

Application Health Bars
Each tool observes only a small slice of a device's state and it's generally difficult to trace the causality of an event across tool boundaries due to different representations and lack of time synchronization.
Instrumentation

- Router and server instrumentation
- Program and library instrumentation

Network
- TCP/IP
- Microcontroller
  - GPIO+Buses
  - Circuit