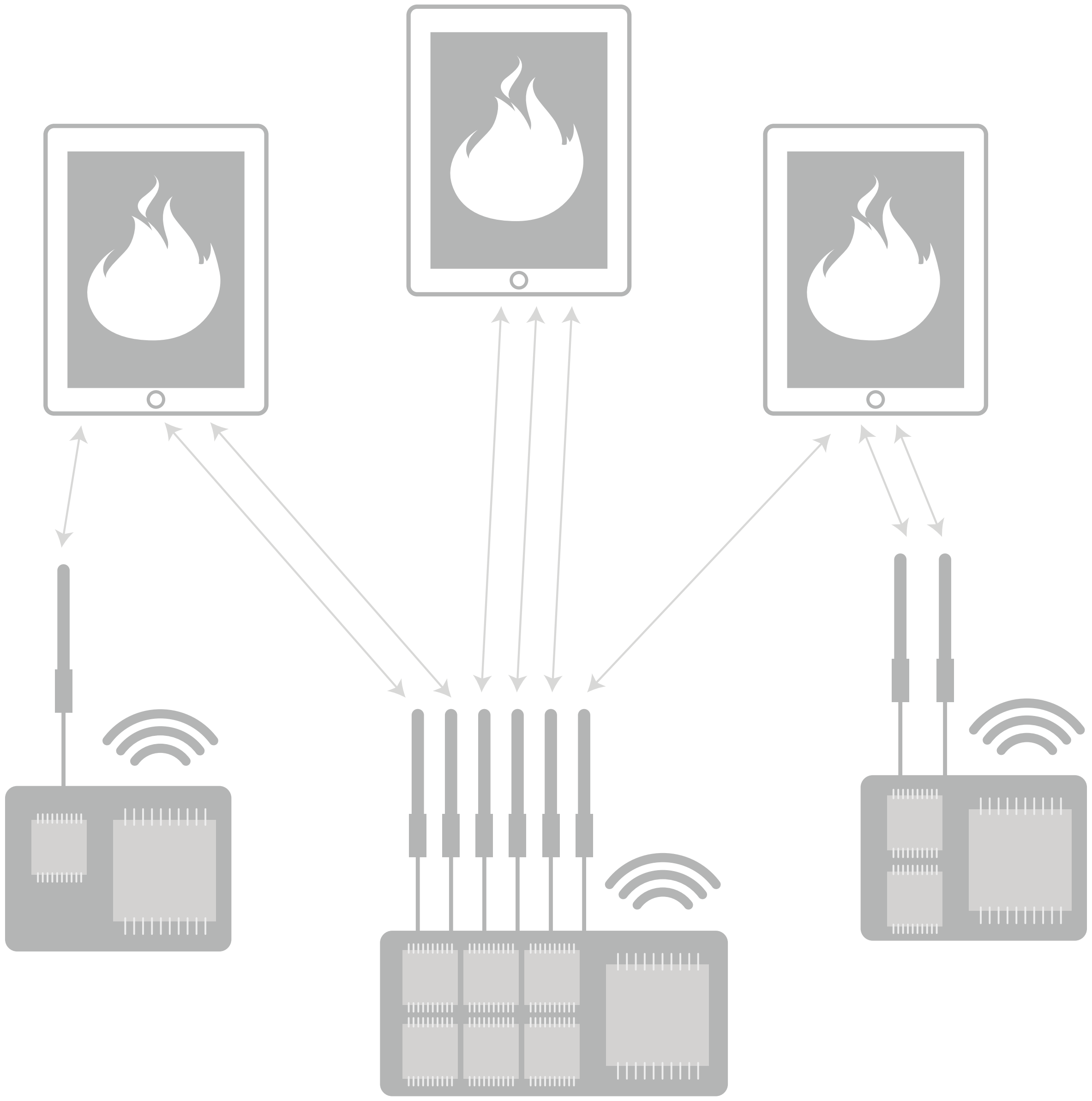
Roastmaster Datagram Protocol (RDP)

Datasheet



Introduction to RDP and Probe Hosts

What is RDP?

RDP is a protocol we created specifically for Roastmaster that allows any processor-equipped circuit board to send temperature data directly to Roastmaster iOS.

Historically, Roastmaster has relied on 3rd party manufacturers to produce both the hardware, as well as the software API that Roastmaster would use to communicate with that hardware. If those APIs contain bugs, or didn't function correctly, there was nothing that could be done with Roastmaster to resolve it.

RDP was specially designed for Roastmaster, and is now built directly into Roastmaster's code. Anyone can use RDP on virtually any hardware. This removes the reliance on 3rd party vendors, and lets anyone design and build a probe host for very little cost that can easily communicate with Roastmaster using this flexible protocol.

It's also fully open source, and published and publicly documented under the permissive MIT license, which means it's free to use by any individual or manufacturer, whether for personal or professional adoption.

What is a Protocol?

A protocol is really nothing more than a language two entities agree to "speak" in order to communicate with each other. If you speak English, you can understand the information in this datasheet because you understand the English language. English is, essentially, the "protocol" of this datasheet.

The same is true with software. As long as two applications or devices speak the same "language", they can talk to each other and share information. Since Roastmaster now "speaks" RDP, any circuit board with a CPU and WiFi capabilities can be programmed to talk directly to Roastmaster using the language of RDP.

What is a Probe Host?

A "probe host" is required to send readings to Roastmaster. A probe host is comprised of the hardware required to gather the data, and simple software to interpret and send this information over WiFi to Roastmaster.

A probe host can be a DIY project, based on a popular SBC (Single Board Computer), such as Arduino, Raspberry Pi, or Feather Huzzah. Or, it can be a professionally designed circuit board built in a roasting appliance, designed and coded by the manufacturer to provide out-of-the-box support for Roastmaster iOS.

Whatever the specifics, as long as it can run software, read a probe, and send those readings as UDP packets over WiFi, it can be used to communicate those readings to Roastmaster iOS.

Networking Basics

UDP Protocol

RDP datagrams are sent using the UDP networking protocol (User Datagram Protocol).

UDP is a simple, uni-directional, streamlined networking protocol, designed for speed, low network overhead, and easy implementation at the programming level. Most programming languages offer simple, one-line calls to send UDP datagrams. It is very easy to write a UDP based program or sketch, without the knowledge and effort it would require for managing the more complex networking sockets of its cousin protocol, TCP.

UDP is, however, a "best effort" protocol. Datagrams are sent to the network, destined for a particlar IP address. The networking topology attempts to deliver them in a timely fashion, but provides no guarantee that they will arrive, or arrive in the correct order. For this reason, it is always best to employ the "Epoch" feature of RDP, to ensure correct packet ordering. It's also advised to send data at least once every second, regardless of whether any temperature data has changed, to guard against the rare case of dropped packets.

Addressing

When sending RDP data, you must supply the correct IP address of the device running Roastmaster in order for it to receive the datagram. You can accomplish this by using either multicast or unicast.

Multicast

This method will deliver datagrams to **every** host on the network by using the special "all hosts" IP address 224.0.0.1. It does not require you to know the IP address of the device running Roastmaster, but will increase network traffic because each transmission is delivered to every computer and device on your network.

This is the easiest way to code your probe host software, since it does not involve any handshaking, and can be implemented in just a few lines of code.

Unicast

This method will deliver datagrams to only one specific IP address. It requires you to know the IP address of the device running Roastmaster.

This requires a little more coding than the multicast method, but reduces network traffic since only one datagram needs to be delivered.

When using the unicast method, it is not advisable to hard code the IP address of the device running Roastmaster. In a typical DHCP network, IP addresses are usually not guaranteed on a per-device basis. When the device is powered off, its IP address will likely change when it is powered back on.

For this reason, a host must ascertain the correct IP address via **Handshaking** before sending RDP datagrams, or risk the probability of sending them to the wrong address.

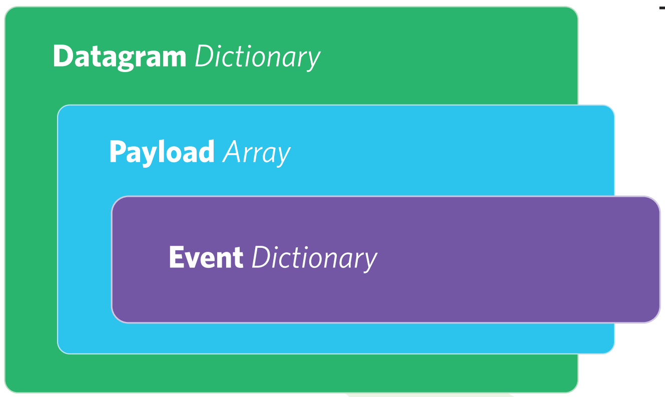
Handshaking

RDP provides a built-in means of learning the target device's IP address through a method called handshaking.

Handshaking is a simple process, in which a probe host sends a multicast RDP datagram with SYN (Sync) information to every device via the all hosts address 224.0.0.1. When Roastmaster detects a SYN request in an incoming datagram, it will reply with a datagram containing ACK (Acknowledgement) information.

To complete the handshaking process, your probe host software should read incoming RDP datagram replies and look for SYN information. If found, it should cache the IP address that the datagram originated from. This will be the IP address of the device running the target copy of Roastmaster, and complete the handshaking process.

Anatomy of an RDP Datagram

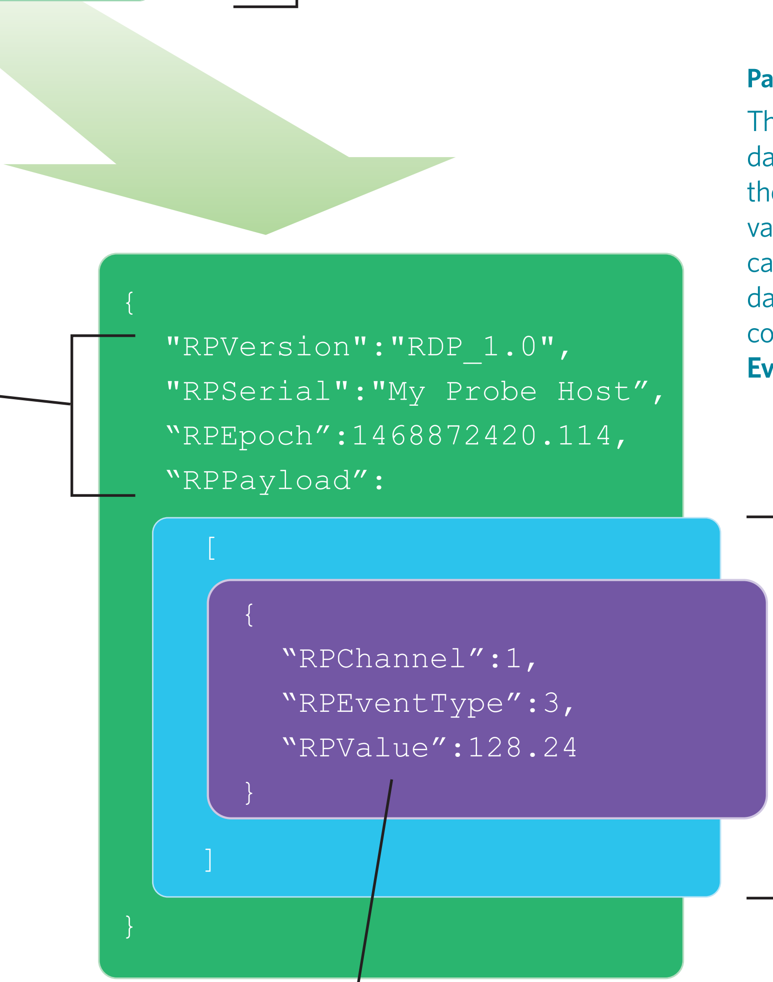


RDP Datagram

The root of an RDP datagram is a JSON dictionary consisting of a series of key/value pairs describing the datagram.

One of these key/value pairs can contain an array, called the payload.

Within the payload array, there can be one or more events–each one a single dictionary consisting of key/value pairs describing that event.



```
{
   "RPVersion":"RDP_1.0",
   "RPSerial":"My Probe Host”,
   “RPEpoch”:1468872420.114,
   “RPPayload”:
      [
         {
            “RPChannel”:1,
            “RPEventType”:3,
            “RPValue”:128.24
         }
      ]
}
```

Header

The key/value pairs of the root datagram dictionary form the **header**. This tells Roastmaster important information about the data contained in the datagram, and how to process it.

Roastmaster first checks the **RPVersion** to see if it is a version it supports, and to determine how to decode the datagram.

Then, it reads the **RPSerial** value to see if it should be interested in any of the datagram’s information, based on the serial fields of the probes defined in Roastmaster.

If so, it examines the value of the **RPEpoch** key. If its value is greater than the last RPEpoch Roastmaster encountered for that serial string (or if no value is supplied) it will process the payload. Otherwise, the packet is ignored.

Correct use of the RPEpoch value will ensure that packets are always processed in the correct order. If an outdated packet arrives with an old epoch value, Roastmaster will simply ignore it.

Payload

The final key in the root datagram dictionary is the RPPayload key. Its value should be an array, called the **payload** of the datagram, and should contain one or more **Event** dictionaries.

Event

When Roastmaster determines that it needs to process a datagram, it will examine the datagram’s payload, looking for individual events.

Events each contain a numerical type, channel and value.

Creating an RDP Temperature Datagram

```
{
   "RPVersion":"RDP_1.0",
   "RPSerial":"My Probe Host”,
   “RPEpoch”:1468872420.114,
   “RPPayload”:
      [
         {
            “RPChannel”:1,
            “RPEventType”:3,
            “RPValue”:128.24
         }
         {
            “RPChannel”:2,
            “RPEventType”:3,
            “RPValue”:204.87
         }
         {
            “RPChannel”:3,
            “RPEventType”:3,
            “RPValue”:37.64
         }
      ]
}
```

Datagram *Dictionary*

Key: **RPVersion** Value: **RDP_1.0** *String*

Key: **RPSerial** Value: *String*

Denotes the serial number (string) that the payload events are destined for. This should match the appropriate probe definition in Roastmaster.

Roastmaster will assign an incoming event’s data to the probe definition whose serial number and channel match that of the event.

You may have one or more hosts, each transmitting information bound for one or more copies of Roastmaster at the same time. The probe definitions in Roastmaster define which events it uses, and which it ignores.

Key: **RPEpoch** Value: *Integer or Double*

Counter or time interval type number supplied by host to ensure correct packet order processing in Roastmaster.

It is easiest to calculate the number of milliseconds since Unix Epoch Time (Jan 1, 1970) and supply this double-precision float. Alternatively, you can use a simple integer counter, so long as the value is > 0.

If included, Roastmaster will ignore packets that have a lesser value than the last one it processed, ensuring correct ordering.

If omitted, Roastmaster will process every packet in the order in which it is received, which is not guaranteed to be the order in which it was sent.

Key: **RPPayload** Value: *Array of Dictionaries (Events)*

Payload *Array of Dictionaries (Events)*

Event *Dictionary*

Key: **RPChannel** Value: *Integer*

Corresponds to the channel defined in the Roastmaster probe definition (1-16).

Key: **RPEventType** Value: *Integer*
(Event Type Constant)

Key: **RPValue** Value: *Float*

The probe reading in Celsius. Roastmaster will translate as appropriate, depending on the measurement system set in the curve using the probe.

Key: **RPMetaType** Value: *Integer*
(Meta Type Constant)

Additional information that can further describe the details of an event. These are currently ignored by Roastmaster, but planned for future implementation.

The Anatomy of an RDP Handshaking Transmission

```
{
  "RPVersion":"RDP_1.0",
  "RPSerial":"My Probe Host”,
  “RPEpoch”:1468872420.114,
  “RPPayload”:
    [
      {
        “RPEventType”:1
      }
    ]
}
```

Synch (SYN) Datagram

Sent by a probe host (usually as a multicast) to obtain a Roastmaster IP address.

Its payload should consist of one event with the SYN Event Type Constant integer.

When Roastmaster receives a SYN request, it resets it Epoch value for that serial number, and sends back a corresponding Acknowledgement (ACK) datagram.

Event *Dictionary*

Key: **RPEventType** Value: *Integer*
(Event Type SYN Constant)

```
{
  "RPVersion":"RDP_1.0",
  "RPSerial":"My Probe Host”,
  “RPEpoch”:1468872420.114,
  “RPPayload”:
    [
      {
        “RPEventType”:2
      }
    ]
}
```

Acknowledgement (ACK) Datagram

Sent by Roastmaster whenever it receives an SYN datagram. One ACK is sent back to the originating address of each SYN packet it receives.

Its payload should consist of one event with the ACK Event Type Constant integer.

The host should read this datagram, store the originating address, using it as the IP address of future datagrams.

Event *Dictionary*

Key: **RPEventType** Value: *Integer*
(Event Type ACK Constant)

RDP Constants Reference

To reduce datagram size, RDP uses integer constants to represent some data types. The following chart lists the constants defined in RDP version 1.0. Roastmaster does not currently utilize RPMetaType, but will as the protocol is expanded.

RPVersion *String Constant*

RDP_1.0

RPEventType *Integer Constant*

1	Sync Request (SYN). Sent from host to Roastmaster.
2	Sync Acknowledgement (ACK). Sent from Roastmaster back to host.
3	Temperature
4	Control
5	Pressure
6	Remote
7	User Defined

RPMetaType *Integer Constant*

3000-3999
Valid Temperature meta types for events with RPEventType 3:

3000	BT Temp
3001	ET Temp
3002	MET Temp
3003	Heat Box Temp
3004	Exhaust Temp
3005	Ambient Temp
3006	BT Cooling Temp

4000-4999
Valid Control meta types for events with RPEventType 4:

4000	Gas Rate
4001	Electric Level
4002	Drum Speed
4003	Fan Speed
4004	Damper Setting

5000-5999
Valid Pressure meta types for events with RPEventType 5:

5000	Drum Pressure
5001	Ambient Pressure
5002	Exhaust Pressure

6000-6999
Valid Remote meta types for events with RPEventType 6:

Currently unpublished

7000-7999
User-defined meta types for events with RPEventType 7:

User defined

Important Notes About RDP

Datagram Frequency

It is advisable to send a datagram containing temps for every probe at least once per second, whether readings have changed or not. This ensures redundancy in cases where UDP packets arrive out of order, or are dropped altogether.

Furthermore, Roastmaster presents a lost link alert if more than 5 seconds elapse between datagrams. A higher datagram frequency ensures against lost links.

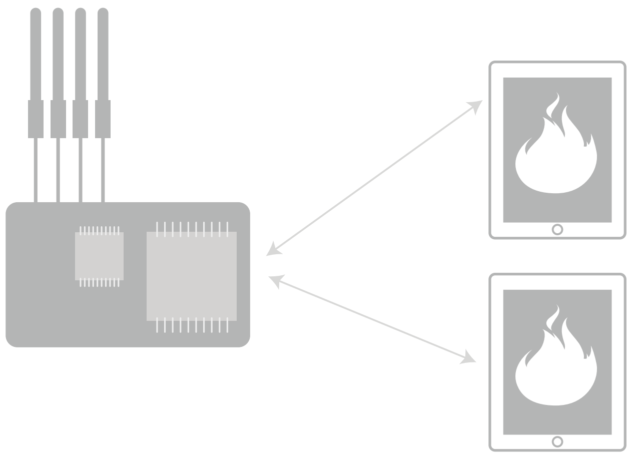
Keep-alive Datagrams

For special cases where it might not be desirable to send a temp in every datagram, RDP supports datagrams without a payload. As long as the header information is present, Roastmaster will reset its internal link timer whenever a datagram is received, whether or not it contains a payload.

Epoch

You are not required to use the RPEpoch header key/value, but it is strongly advised. It you omit this, Roastmaster will process every packet in the order in which it is received, which is not guaranteed by UDP to be the original, correct order. Using an epoch value that is incremented after every datagram transmission ensures that packets are processed in the correct order. If an old packet arrives, Roastmaster simply ignores it.

Zoning



A probe host can contain multiple thermocouples. Furthermore, certain thermocouple amp boards can be wired in series, allowing a very large number of thermocouples to be used on the same host.

Since each RDP serial can support up to 16 logical probe ports, simple zoning can be accomplished via the “Port” value of RDP events.

For example, you might have two iPads running Roastmaster. The first would define probes on ports 1 and 2, and the second would define probes on port 3 and 4–all with the same serial. The probe host would send all four probes in its datagram. Roastmaster would only process the events applicable to itself.

It is also possible to zone a single probe host using multiple “Serial” values. In this case, different instances of Roastmaster would define probes with different serial values. The probe host would formulate different datagrams for each serial value, according to its own zoning requirements.

Both of these zoning examples require A) maintaining an array of IP addresses via RDP handshaking instead of just one IP address, or B) utilizing the multicast method.

Formatting JSON

Creating an RDP datagram for WiFi transmission to Roastmaster is a very easy process.

RDP packets are formed using JSON (JavaScript Object Notation). JSON is a means of expressing programming objects (such as strings, numbers, dictionaries and arrays) with a simple string. The notation used determines the object types that are represented.

```
{
  "RPVersion":"RDP_1.0",
  "RPSerial":"My Probe Host”,
  “RPEpoch”:1468872420.114,
  “RPPayload”:
    [
      {
        “RPChannel”:1,
        “RPEventType”:3,
        “RPValue”:128.24
      }
    ]
}
```

Strings

Strings are expressed within quotes, e.g. “Bean”, “Blend”, etc.

Numbers

Numbers are expressed as numerals without quotes, e.g. 2.8, 7, etc.

Arrays

Arrays are expressed as comma separated values, surrounded in square brackets, e.g. [“Apple”,2,4,“Pear”]

Dictionaries

Dictionaries are expressed as a series of comma separated key/value pairs, surrounded in parentheses. Each key/value pair begins with the key (a string), followed by a colon, and then its value., e.g. {“Name”:”Robert”, “Age”:21}.

Hierarchical Items

JSON supports hierarchical information, so you can embed arrays inside dictionaries or dictionaries inside arrays.

RDP uses this feature to construct a header/payload format, and to support multiple events within a single payload.

There are many JSON libraries available for most programming languages, designed to automatically convert programming structures directly to JSON. But, since an RDP datagram is very simple, it’s usually easiest to construct the format yourself in code, using simple string concatenation.

Pretty-Formatting should be avoided

For the purpose of legibility for this datasheet, RDP examples are pretty-formatted, with indentation and paragraph returns. However, JSON is rarely pretty-formatted in practice.

In code, the example at the top lift would appear as a continuous string, devoid of indents and returns:

{"RPVersion":"RDP_1.0","RPSerial":"My Probe Host”,“RPEpoch”:1468872420.114,“RPPayload”:[{“RPChannel”:1,“RPEventType”:3,“RPValue”:128.24}]}

License

Written by Danny Hall, for Rainfrog, Inc.

Special Thanks

Based on input from countless Roastmaster users.

Evan Graham - Thanks for your incredibly imaginative solution to a seemingly insurmountable enigma by developing the first working prototype for getting digital readings from a rotating Gene Cafe drum

Robert Swift - Thanks for your impetus, tireless work, vision and code prototyping

Terms of Use

Using, developing, sharing, distributing, selling, promoting, or any other action involving the Roastmaster Datagram Protocol (RDP), and associated software and documentation, whether personal or professional, binds you to the following license.

MIT License