Kegboard Documentation

Release 0.0

mike wakerly

Jul 21, 2022

CONTENTS

This document describes how to program and hack on a Kegboard, our Arduino-based keg controller board.

Kegboard software works with the Kegboard Pro Mini, and can also be used with a do-it-yourself Arduino Kegboard.

CHAPTER

ONE

KEGBOARD OVERVIEW

This page describes *Kegboard*, the Arduino-based controller board for Kegbot.

1.1 What is a Kegboard?

Kegboard is the name we use for the microcontroller board used in a Kegbot system. Kegboard is the device that monitors all sensors, including the flow sensors that are essential in any Kegbot configuration.

There are two commonly-used Kegboard targets:

- Kegboard Pro Mini, a fully-assembled board, introduced in early 2014 and sold at the Kegbot Store.
- DIY Arduino Kegboard, a do-it-yourself option that can be built using an Arduino board.

The Kegboard software package includes firmware and support libraries that work with either kind of Kegboard.

1.2 Features

Since not all Kegbots are alike, the Kegbot firmware is designed with flexibility in mind: We try to support many features and add-on devices in the core firmware, while still keeping basic functionality tight and fast for the common configurations.

Depending on hardware, the Kegboard firmware can support the following features:

- **Flow Sensing:** Two independent flow meter inputs (or 6 on Arduino Mega), allowing you to monitor that many individual beer taps with just one board.
- **Temperature Sensing:** Dedicated OneWire bus for reading DS1820 (DS18S20 and DS18B20) temperature sensors. An unlimited number of sensors can be connected, allowing you to independently track keg temperature and ambient temperature.
- **RFID Authentication:** Authenticate users with cheap 125kHz RFIDs by connecting the optional ID-12 RFID reader.
- **OneWire Authentication:** Authenticate users with durable iButtons.
- **Relay/Value Control:** Four general purpose outputs can be used to toggle external devices, such as a valve to prevent unauthorized access. Relays are monitored by an internal watchdog.
- **Buzzer:** Kegboard will play a short melody whenever an authentication token is connected or swiped.
- **Extensible Serial Protocol:** If you don't want to use the rest of the Kegbot software, you can still use Kegboard by implementing its simple and extensible serial protocol in your system. (See *Kegboard Serial Protocol Reference*).

Note: Because of its limited size, certain features (such as relay control and RFID reading) are not available on Kegboard Pro Mini.

Kegboard's firmware is designed to operate correctly even when a feature is not being used. For example, if the temperature sensor input is not connected, other features will continue to operate normally.

CHAPTER

TWO

THEORY OF OPERATION

This chapter describes the internal design of the Kegboard firmware and how it manages connected sensors. If you're not interested, you can safely skip to the next chapter.

2.1 Main event loop

Kegboard's two principle responsibilities are:

- Monitors and report status and events from attached sensors.
- Accept commands from the host to enable and disable output relays.

When the board is powered, it immediately begins listening to sensors and sending events on the serial port. If temperature sensing is enabled, the board also periodically polls attached sensors. Additionally, the host can send commands the board at any time. (Commands and events are detailed in *Kegboard Serial Protocol Reference*.)

2.2 Flow sensing

Each flow meter is connected to one of the Arduino's external interrupt pins. (On an Arduino, these are digital pins 2 and 3.)

Kegboard supports "open collector" flow meters. These meters are typically built using hall effect sensors. As liquid passes through the meter, a series of pulses is emitted on its output pin.

Every pulse emitted by the meter corresponds to the same fixed volume of fluid, therefore volume is determined simply by counting the pulses. The exact volume of a pulse is a physical property of the meter; the popular Vision 2000 meter pulses 2200 times per liter.

In the interrupt service routine for each of these pins, Kegboard increments a counter every time there is a pulse, keeping a running total of each meter's volume (similar to an odometer).

2.3 OneWire presence and temperature sensing

The Kegboard firmware supports two distinct OneWire (1-wire) sensor busses: the "thermo" bus, and the "presence" bus.

The "thermo" bus supports reading Dallas/Maxim DS18B20 and 18S20 OneWire temperature sensors. This bus is reserved exclusively for temperature sensors; OneWire devices not matching the DS18B20 or DS18S20 family codes will be ignored on this bus. Any number of sensors may be attached.

The firmware also supports a second OneWire bus, which is continuously polled for OneWire devices. Whenever a OneWire device such as an iButton is connected, its unique 64-bit OneWire device ID is reported as an authentication token using *Kegboard Serial Protocol Reference*.

2.4 Relays

When a relay is enabled, Kegboard enables the corresponding output and starts a timer. If the host has not re-activated the relay within that timer, Kegboard automatically deactivated the output. This prevents prolonged relay operation if the host crashes unexpectedly.

2.5 Piezo buzzer

A low-cost piezo buzzer can be connected to the *buzzer output pin*. When connected, Kegboard will serenade you with some sweet tunes.

Event	Sound
Board power up	Short musical tune (4 notes).
Auth Token Added	Three-tone “added” sound.

CHAPTER

THREE

PIN MAPPING

The following table describes the pins available on a standard Arduino board, and their purpose in the Kegboard firmware.

Pin	Description
2	Flowmeter 0 input
3	Flowmeter 1 input
4	Flow 0 LED output
5	Flow 1 LED output
6	ID-12 RFID input
7	Thermo OneWire bus
8	Presence OneWire bus
9	GPIO pin C
10	ID-12 RFID reset output
11	Buzzer output
12	Test pulse output
13	Alarm output
A0	Relay 0 output
A1	Relay 1 output
A2	Relay 2 output
A3	Relay 3 output
A4	GPIO pin A
A5	GPIO pin B

CHAPTER

FOUR

BUILDING AND FLASHING FIRMWARE

This section describes how to build and flash the Kegboard firmware on a standard Arduino device.

4.1 Install Arduino software

The Arduino project provides a free software development environment for Mac, Windows, and Linux. It includes a basic text editor, avr microcontroller toolchains, and many standard libraries. In short, it is everything you need to program an arduino board.

Download the software from the Arduino Downloads Page. Packages are available for Linux, Mac OS X, and Windows.

Note: The latest tested version is Arduino 1.0.4

When unzipped you will have a single directory that contains all the arduino software. The name will be something like `arduino-1.0.4/` (the version number will be different for previous versions.)

Place this directory somewhere appropriate. For Mac users, you can drag it to your Applications folder.

4.2 Compile and flash the firmware

A new Arduino board includes a basic bootloader on internal flash. The board needs to be programmed with the custom Kegboard firmware.

Binary versions of the Kegboard firmware are not provided, so you need to build it yourself. This process isn't too hard. If you already have the Arduino software installed, and you have a clone of the kegboard repository somewhere, you're most of the way there

The latest version of the Kegboard firmware is available in the **kegboard** distribution, under the directory `src/kegboard/`.

You can also download the entire Kegboard github repository as a zip file: Download Kegboard repository.

The file `kegboard.ino` is the main source to the firmware. This file is a C source file, using the file extension preferred by the Arduino development tools.

4.2.1 Compile

Open the file `kegboard.ino` in the Arduino studio application. You should see a listing of the source. You do not need to make any changes to the source.

Next, configure the Arduino environment to match your Arduino. In particular:

- Select the correct board type from menu *Tools* → *Board*
- Select the serial port it is attached to from the menu *Tools* → *Serial Port*

You now be ready to build the firmware. Select the menu item *Sketch* → *Verify/Compile*.

4.2.2 Flash

To install the firmware, should select the menu *File* → *Upload to I/O Board* in the Arduino software. The firmware will be uploaded to you device.

Depending on your hardware, it may be necessary to reset the board using the reset pushbutton when starting the upload.

4.2.3 Test pin

To simulate a flow meter, you can connect Pin 12 to either of the two flow meter pins with a short jumper wire. This pin continuously outputs a slow stream of pulses, much like a flow meter would do.

CHAPTER

FIVE

KEGBOARD SUPPORT LIBRARY

In addition to the Kegboard firmware, the Kegboard package comes with some support optional tools (`kegboard-monitor` and `kegboard-tester`) as well as a Python library for writing new programs to talk to Kegboard. This section will walk you through installing them.

Note: The support tools are automatically installed as part of Pycore. If you've installed Pycore, you can skip this section.

5.1 Set up virtualenv

Note: If you've already installed Kegbot Server or Pycore in its own virtualenv, you don't need to create a new one just for Kegbot; it's perfectly fine to reuse the existing virtualenv.

The `virtualenv` tool creates a directory where the Kegboard support programs and all of their Python dependencies will be stored. It makes it easier to install the tools without needing root privileges, and reduces the chance of Kegboard clashing with your system's Python modules.

The first time you set up Kegboard, you will need to create a new virtualenv "home" for it. Any filesystem location is fine. To create it, give the directory name as the only argument. The example below creates the Kegboard virtualenv directory in your user's home directory:

```
$ virtualenv ~/kegboard
New python executable in /Users/mike/kegboard/bin/python
Installing setuptools.............done.
Installing pip...............done.
```

Now that the virtualenv home has been created (at `~/kegboard/`), there's one step to remember. Each time you want to use the virtualenv home (to use the support tools) you need to activate it for the current shell:

```
$ source ~/kegboard/bin/activate
(kegboard) $
```

Your shell prompt will be updated with `(kegboard)` when the virtualenv is active. If you want to step out of the env for some reason, just call `deactivate`:

```
(kegboard) $ deactivate
$
```

If you ever want to completely uninstall Kegboard, just delete the entire `kegboard/` directory – there's nothing precious in it, and you can always recreate it by following these steps again.

Tip: You can install multiple versions of Kegboard simply by creating a new virtualenv for each one.

5.2 Install Kegboard

There are two approaches to downloading and installing Kegbot:

- *From the latest release, using pip*, the recommended way to quickly get going with the latest release.
- *From Git*, to grab the latest, bleeding-edge development code. Recommended for advanced users only.

If in doubt, proceed to the next section for the easiest method. Be sure you have activated your virtualenv first.

5.2.1 From Latest Release (Recommended)

Use the `pip` tool to install the latest release of Kegboard, including its dependencies:

```
(kegboard) $ pip install kegbot-kegboard
```

The command may take a few minutes as it downloads and installs everything.

5.2.2 From Git (developers)

If you want to run Kegboard from the latest development version, follow this section.

Warning: Running from unreleased git sources is not recommended for production systems, since code can be very unstable and functionality may change suddenly. **Always** back up your valuable data. As stated in Kegbot's license, we provide Kegbot with absolutely no warranty.

1. Check out the Kegboard sources using `git`:

   ```
   (kegboard) $ git clone https://github.com/Kegbot/kegboard.git
   ```

2. Step in to the new tree and run the setup command:

   ```
   (kegboard) $ cd kegbot/python
   (kegboard) $ ./setup.py develop
   ```

The command may take a few minutes as it downloads and installs everything.

5.3 Testing a Kegboard

The support library includes two small programs which you can use against a connected Kegboard.

5.3.1 kegboard-monitor.py

This program monitors your system's serial ports for a Kegboard, displaying information about each one it detects.

5.3.2 kegboard-tester.py

This program cycles through each relay on the Kegboard (when available), opening and closing it.

CHAPTER

SIX

KEGBOARD SERIAL PROTOCOL REFERENCE

6.1 About

This document describes the protocol implemented in the Kegboard firmware. The protocol is a simple serial protocol for exchanging data and commands between a host computer and the controller board.

Note: Most users don't need to be too familiar with the Kegboard protocol. This document is intended for someone building a new type of controller board, or attempting to extend the existing board. For example, if you build a new type of controller board that speaks this protocol, you should be able to use the rest of the Kegboard software without further modification.

6.2 Protocol Overview

The Kegboard Serial Protocol is a binary protocol between the kegboard and the host PC. Data is delivered from the board in the **message frame format** (described later). The host can also control and configure the board by sending messages in the same format.

Messages arrive from the board *asynchronously*; the host does not need to request updates to receive information about sensors. Similarly, the host can issue commands to the board at any time.

6.3 Software Support

The pykeg software includes libraries for reading and writing KBSP packets. A unittest with a sample packet capture is also included. See the code available in `pykeg/hw/kegboard`.

6.4 Message Frame Format

Data from the Kegboard is always sent to the host in the *Kegboard Message* frame format. All messages take the same basic format:

```
<header><payload><footer>
```

A single message includes:

- A 12-byte header section, identifying the message being sent.

- A variable-length payload section, typically in TLV (tag-length-value) format. The payload can be from zero to 112 bytes in size.
- A 4-byte footer section, containing a CRC of the entire packet.

Given the section sizes above, the maximum length of a complete message is 128 bytes.

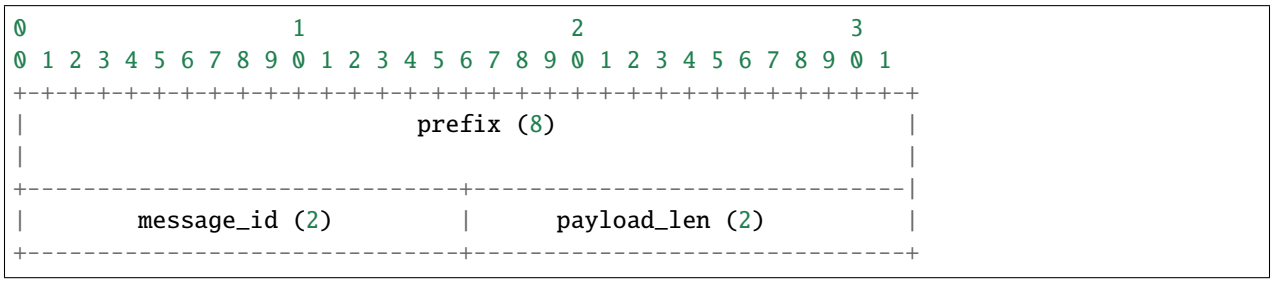
Here are examples of two messages in a raw byte stream, showing two frames sent from the Kegboard to the host. For each message, the first line is a human-readable version of the message; the second line is a sequence of bytes forming the complete message.:

```
# <HelloMessage: protocol_version=3>
\x4b\x42\x53\x50\x20\x76\x31\x3a\x01\x00\x04\x00\x01\x02\x03\x00\x2e\x54\x0d\x0a

# <MeterStatusMessage: meter_name=flow1 meter_reading=4>
\x4b\x42\x53\x50\x20\x76\x31\x3a\x10\x00\x0e\x00\x01\x06\x66\x6c\x6f\x77\x31\x00\x02\x04\
↪x04\x00\x00\x00\x55\x0a\x0d\x0a
```

6.4.1 Header Section

Every message begins with a fixed-length 12-byte header section. An ASCII representation is shown below:



The *prefix* field is a constant 8-byte sequence to identify the start of the packet. The value is always the ascii characters "`KBSP v1:`". Software attaching to a live kegboard serial port can search for this stream of characters to determine the start of the next packet, with reasonable confidence.

The *message_id* field is the type of the message (see types defined in *Message Definitions*.)

The *payload_len* field is the length, in bytes, of the payload section. Some messages do not have a payload, in which case this section will read zero. Note that this value only describes the length of the payload section (and not the length of the footer section, which is constant.)

6.4.2 Payload Section

The payload of a message varies depending on the message type, described in *Message Definitions*. The payload section may be empty, and has a maximum length of 112 bytes.

All payloads are serialized in type-length-value style. This format makes it possible to extend the body of a payload in the future with a new tag number. If a message parser encounters a tag it does not recognize, it should skip over the unrecognized value according to the given length.

6.4.3 Footer Section

Each message includes a 4-byte footer section:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           crc (2)             |         trailer (2)           |
+-------------------------------+-------------------------------+
```

The *crc* field is a CRC-16-CCITT of the header and payload fields; in other words, the entire message up to the CRC. This CRC is used by the host to verify the integrity of messages from the board.

The integrity of a message can be verified by performing the CRC calculation on all data, up to and including the CRC (but not the trailer). If correct, the value will be zero.

The string `\r\n` is always written in the *trailer* field. This field is not included in the CRC.

6.5 Field Types

Fields in messages are described in terms of *field types*, which are analogous to C types. Types used are described below. Note that all integer types are serialized in **little-endian** format.

Type name	Size	Interpretation
`int8`	1	8-bit signed integer (aka char)
`int16`	2	16-bit signed integer
`int32`	4	32-bit signed integer
`uint8`	1	8-bit unsigned integer (aka uchar)
`uint16`	2	16-bit unsigned integer
`uint32`	4	32-bit unsigned integer
`uint64`	8	64-bit unsigned integer
`string`	Varies	Null-terminated C string
`bytes`	Varies	Raw collection of byte values.
`output_t`	1	Boolean (0=disabled, 1=enabled); like `uint8`
`temp_t`	4	1/10^6 Degrees C; signed; like `int32`

6.6 Message Definitions

This section summarizes messages which may arrive at the host from a board implementing the protocol.

6.6.1 `hello` message (0x01)

This message may be sent by the board to indicate that it is alive. The host may request this message with the *ping command (0x81)*.

Tag ID	Name	Type	Description
0x01	firmware_version	uint16	Board firmware version.
0x02	protocol_version	uint16	Supported version of kegboard serial protocol.
0x03	serial_number	string	Board serial number

6.6.2 board_configuration message (0x02)

A configuration message dumps the board's configuration data. These values can be programmed by sending a set-configuration-command with the same message as payload.

Tag ID	Name	Type	Description
0x01	board_name	string	Board descriptive name.
0x02	baud_rate	uint16	Serial port speed, in bits per second
0x03	update_interval	uint16	Time in milliseconds between update messages to the host.
0x04	watchdog_timeout	uint16	Maximum time permitted between commands from host before triggering the watchdog alarm.

6.6.3 meter_status message (0x10)

This message describes the instantaneous reading of a single flow meter channel. For a kegboard with multiple flow meter inputs, multiple messages will be sent.

Tag ID	Name	Type	Description
0x01	meter_name	string	Name of the meter reporting flow.
0x02	meter_reading	uint32	Total volume, in ticks.

6.6.4 temperature_reading message (0x11)

This message describes the instantaneous reading of a single temperature sensor. For a kegboard with multiple sensors, multiple messages may be sent. Note that the temperature is presumed to be valid at the time the message is sent.

The value of `sensor_name` will include the full 128-bit 1-wire device id, for example, `thermo-f800080012345610`.

Tag ID	Name	Type	Description
0x01	sensor_name	string	Name of the sensor being repoted.
0x02	sensor_reading	temp_t	Temperature at the sensor.

6.6.5 output_status message (0x12)

This message describes the status of a single general-purpose output on the board. An output could be connected a relay, or some other device to control valves.

Tag ID	Name	Type	Description
0x01	output_name	string	Name of the output being reported.
0x02	output_reading	output_t	Status of the output.

6.6.6 `auth_token` message (0x14)

When an authentication token is attached or removed from the kegboard, this messages is sent. The `device_name` field gives the name of the kegboard peripheral producing the message; this will be *onewire* for iButtons and `rfid` for RFIDs. The `token` field gives the raw, big-endian byte value of the token.

Tag ID	Name	Type	Description
0x01	device_name	string	Name of authentication device.
0x02	token	bytes	Raw token ID being reported.
0x03	status	uint8	1 if present, 0 if removed.

6.7 Command Definitions

This section summarizes messages which may be sent to a host implementing the protocol.

6.7.1 `ping` command (0x81)

This command is sent to the board to request a *hello message (0x01)*. This can be used to verify that the attached device is a Kegboard that speaks the serial protocol.

There is no payload.

6.7.2 `set_output` command (0x84)

This command is sent to the board to enable or disable a device output.

Tag ID	Name	Type	Description
0x01	output_id	uint8_t	Numerical output id (0-15).
0x02	output_mode	output_t	Mode to set the output.

6.7.3 `set_serial_number` command (0x84)

This command sets the board serial number, if not already set.

Tag ID	Name	Type	Description
0x01	serial_number	string	Serial number.

6.8 Protocol Revision History

This section describes major updates to this protocol.

Version	Date	Remarks
1	current	Initial version.

CHAPTER

SEVEN

CHANGELOG

7.1 Arduino Firmware

7.1.1 v18 (2014-05-14)

- Added heartbeat: device will send a Hello message every 10 seconds.
- Hello message now includes uptime information.

7.1.2 v17 (2014-02-12)

- Fixed a bug that broke serial communication (introduced in v16).

7.1.3 v16 (2014-02-07)

- Added chip LED support for Kegboard Pro Mini.

7.1.4 v15 (2014-01-16)

- Added *set_serial_number* command.

7.1.5 v14 (2013-07-23)

- Flow LEDs are now toggled on system startup and during pours.
- Experimental debounce feature.
- Support for Parallax RFID readers.

7.1.6 v13 (2012-10-28)

- Adds support for Wiegand RFID readers (HID ProxPro and similar).

7.1.7 v12 (2012-07-07)

- Respond to ping with a short melody.

7.1.8 v11 (2012-05-02)

- Updates for Arduino SDK v1.0; no functional changes.

7.1.9 v10 (2011-06-19)

- Reverse ID-12 RFID endianness.

7.1.10 v9 (2011-06-13)

- Support ID-12 RFID input

7.1.11 v8 (2011-06-11)

- Expand 'set_output' to support onboard kegboard relay's, flow led's

7.1.12 v7 (2011-03-16)

- Added implementation of *set_output* command, relay output watchdog.

7.1.13 v6 (2010-09-22)

- Added auth_token message.

7.1.14 v5 (2010-01-10)

- Fix issue that caused flow events to be reported too frequently.

7.1.15 v4 (2010-01-04)

- Initial documented version.

7.2 Support Library

7.2.1 v1.0.0 (2012-07-01)

- Support library added to kegboard repository.
- Previous versions were located in the old master Kegbot repository: https://github.com/Kegbot/kegbot