AndroHART Burst Slave documentation

Kovács Levente

This app (AndroHART Burst Slave) is capable of simulating a field device by sending command 01, 02, or 03 from universal commands periodically, in burst mode. This is done through the HART protocol, and – on a physical level – through USB and the connected HART modem.

Installation

Prerequisites

To run this program, you must own an Android smartphone, which is capable of OTG functionality, in other words, can function as an USB host. To test this, various free applications are available, but most newer phones support this. This app requires a minimal Android version of 4.2 (Jelly Bean). On phones with Android version under 4.2 this app won't run. Android version can be checked at system settings.

You also need a USB/HART modem to communicate properly. This app was tested with the following:

- Nivelco unicomm SAK-305
- Nivelco Hart-USB modem SAT-304-0
- ESH232U modem

Installation

Acquiring this app from Google Play, it will be installed immediately. It will detect on start, whether is a USB slave connected to it, and if requirements are met, the permission request will be sent. If you plug the USB in later, while the app is already running, it will handle that as well.

Usage

The application consists of tabs, and each tab serves different functionality. You can switch tabs by touching them, while a status indicator icon will always be visible on the menu bar.

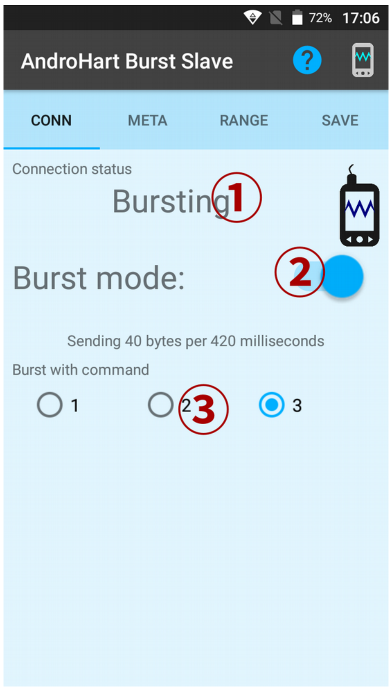
CONN tab

This tab show the status of the connection, also you can turn the burst mode on and off here.

Show status (1)

Status is shown on text and with an icon. The serial port has four statuses (five with bursting):

- detached
- bursting
- available
- no permission granted
- not supported



Connection status
Bursting

Burst mode: ●

Sending 40 bytes per 420 milliseconds

Burst with command
○ 1 ○ 2 [✓ 3]

In the latter case USB vendor and product IDs are also shown, and sending them to the developer might help in future support for these devices.

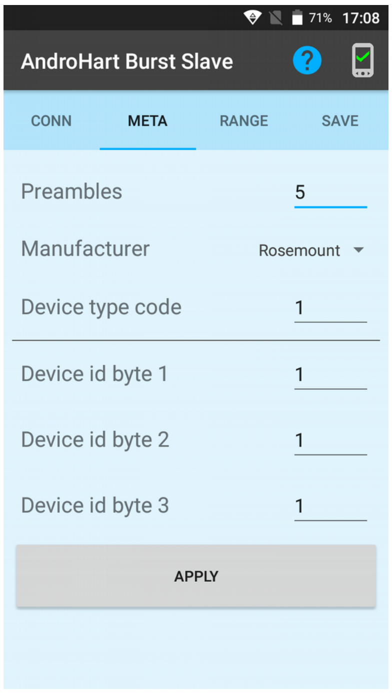
Turning burst on and off (2)

With the marked switch, you can turn the burst mode on and off. Burst mode will only be activated, when the serial port is available, and if you unplug the USB cable, burst mode will also stop. Unplugging while bursting won’t cause any problems to the phone or the app.

Selecting the command to burst (3)

By clicking the radio buttons, you can switch between the command the device will send. Command 3 is the default as it is the most commonly used, but you can send command 1 (only the primary value with its unit type) or 2 (raw and percentage value of the current). Selecting a different command will cause burst mode to exit (you have to restart it manually) due to timing related stuff in the program.

META tab



Preambles	5
Manufacturer	Rosemount
Device type code	1
Device id byte 1	1
Device id byte 2	1
Device id byte 3	1

On this tab, the various meta informations of the device can be set. Some measurement and data registering systems wont care about these props (they’ll only care about the variables sent), although they are required for correct communication (according to the HART specification). Properties, that can be set:

- Preamble count: This is more like of property of the communication and not of the device. This value determines how many 0xFF bytes will be sent at the beginning of the message. In most cases you shouldn’t change this, and you definitely shouldn’t set this to a value lower than 3.
- Manufacturer: The value of manufacturer can be picked from a dropdown list with 256 elements. Not all elements

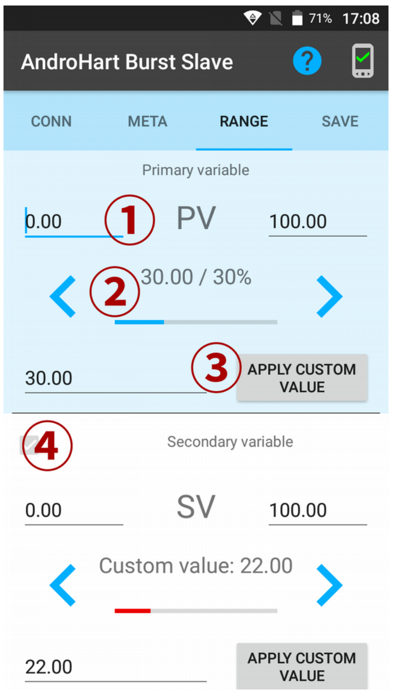
belong to a valid manufacturer, some of them are reserved. The list of manufacturers were acquired from a source, which may have changed since then.

- Device codes: The authentication code of the device. They rarely have any meaning and if they have, it depends on the manufacturer and the subtype of the device. These values can be read from real devices, but you can set random values as well, and as default in this program, each value is set to 1.

You can apply these values by clicking the Apply button. Each value is updated in this case, so each textfield should have a valid value. After a successful update, a message will be shown.

RANGE tab

You can set the values and the ranges of the four dynamic variables here on this tab. The interface for setting the four variables are pretty much the same. You have to set a value for LRV and URV, and by them you can adjust the value of the actual variable. The value of current is also calculated from these limits and values.



Measurement limits (1)

By standard, values can be set by moving a slider in an interval. That means, that not only the values themselves, but the intervals they belong to can be set as well. On the left text field, the value of LRV can be specified, while the text field on the right belongs to URV. To apply these values, press one of the arrows below them, since this updates the value as well. Some basic checking is made, so you cant have an URV smaller than the LRV value.

Displaying the value (2)

The text field and the colorful bar are both act as indicators for the value. By pressing the blue arrows, you can adjust the value in the range of (AMH, FMH) by 10%. Pressing these arrows apply the change instantly, so the new values will be sent is the next burst cycle.

Setting custom value (3)

In case you don’t want to work with intervals, you can set custom values for the variables as well. By pressing the “Apply custom value” button, you can apply the custom value. In that case, it will be shown differently in the text representation and the progress bar will turn red (as shown in SV). By pressing the arrows again, the standard mode will be restored.

Activating variables (4)

When working with command 3, a device is not obligated to send all of its four dynamic variables, since this behavior is allowed by the HART specification. By ticking these marked checkboxes, you can set which variables you want to send. Primary variable will always be sent (therefore it does not have a checkbox next to it), and you can’t leave “holes” in the selection (you can’t send the third variable without sending the second for example). Luckily, you cant really commit those mistakes, since the checkboxes are disabled, when they shouldn't be used (int this example, the marked checkbox is disabled, since the third variable is turned on).

SAVE tab

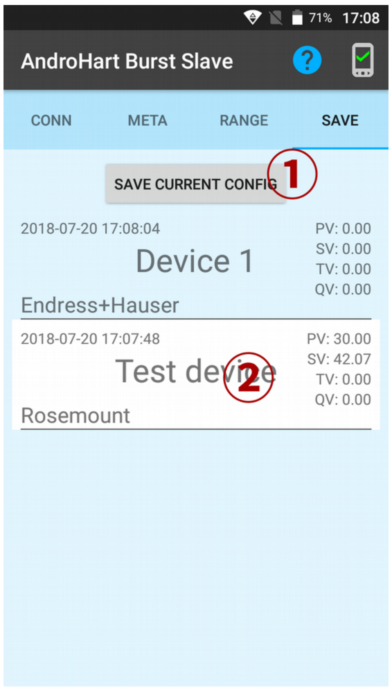
You can load and save existing configurations on this tab. You can set a name for each configuration, though they are not exactly filenames.

Saving current config (1)

Saves the actual config into storage (a database to be precise), so they won't get lost, when the app is closed or the phone is restarted. When saving the config, you can give it a unique name, and in case that name already exists, a prompt about overwriting will be shown. Canceling those dialogs will cancel the save itself.

Saved configurations (2)

The saved configurations are listed below the button. The currently used (even if its modified manually) configuration has a white background. By touching them, you can load that config, and by swiping them to left or right, you can delete them. A confirmation dialog will be shown before executing those actions.



2018-07-20 17:08:04
Device 1
Endress+Hauser
PV: 0.00
SV: 0.00
TV: 0.00
QV: 0.00

2018-07-20 17:07:48
Test device
Rosemount
PV: 30.00
SV: 42.07
TV: 0.00
QV: 0.00

Currently not included

These functions are not included in my program, though the can be implemented on request.

Error and status codes

When communicating through Hart, two bytes are reserved for error and status in each response (burst is a response in that case). These bytes are wired to a constant zero value (which means everything is all right), however, they could be set on the user interface, either by writing the byes in directly or by ticking each bit in a bitfield (in case it should be interpreted as a collection of flag bits).

Value types / Unit codes

Each variable uses 4 bytes of space, however, a fifth is always sent with them, which represents the value type of that variable. To decrypt which byte value corresponds to which value type, predefined lists do exists, but nothing prevents a device from not following them. The values types are constant values currently, and this might or might not be a problem, whether the measurement system checks for them. The list of unit codes from AndroHART can be implemented here as well.

Fix current

While the values of the variables can be set, the value of current is always calculated by interpolation based on the limits and value of the primary variable (in the range of 4-20 mA). On the other hand, if the field device has a poll address other than zero, it sends constant (usually) 4 mA as current, and this behavior is also not implemented in this app, but could be if requested.

Used libraries

- UsbSerial (https://github.com/felHR85/UsbSerial)
- Apache Commons

Versions

- 1.0.0: Initial release
- 1.1.0: Added save
- 1.1.1: About menu, localization
- 1.1.2: Bugfix
- 1.1.3: Added option to change bursting comand
- 1.1.4: Bugfix (fixed command length)