AndroHART manual

Program and manual written by kovlev

Table of Contents

1. Introduction

AndroHART is an application which runs on Android mobile devices. The app is capable of communicating with intelligent field devices through HART, and configure them. In order to set up proper communication, the user must own a wired HART modem and an OTG (USB male – microUSB female) cable.

There are many software available for Pcs, which provide HART configuring functionality on a wider scale, however, the most basic commands are implemented in AndroHART, and for the end user, it might be more convenient to carry around a mobile phone instead of a heavier, bulkier laptop.

2. Installation, usage

2.1. Pre-requirements

To run the program, you must have an Android smartphone with OTG functionality, in other words, the smartphone can be used as an USB host. You can check this with many free programs, however most newer devices support this. AndroHART was compiled with SDK level 17 as minimum SDK level (Android 4.2 JellyBean). It is almost certain, that the app will not run on earlier verisons of Android. The compile level SDK was 26 (Android 8.0 O).

2.2. Download, install

The app is available on and can be downloaded from Google Play. This handles the installation process. On some mobile devices, you must connect the OTG cable before starting the app, however it is not common. If you disconnect the cable before quitting the app, there is a small chance, that the app will crash. It depends on the mobile device and the USB to serial chip.

3. Commands

AndroHART uses multiple tabs, where graphical and logical elements are grouped by HART functionality. The tab view is placed under the header of the application, and can be scrolled horizontally.

3.1. Header

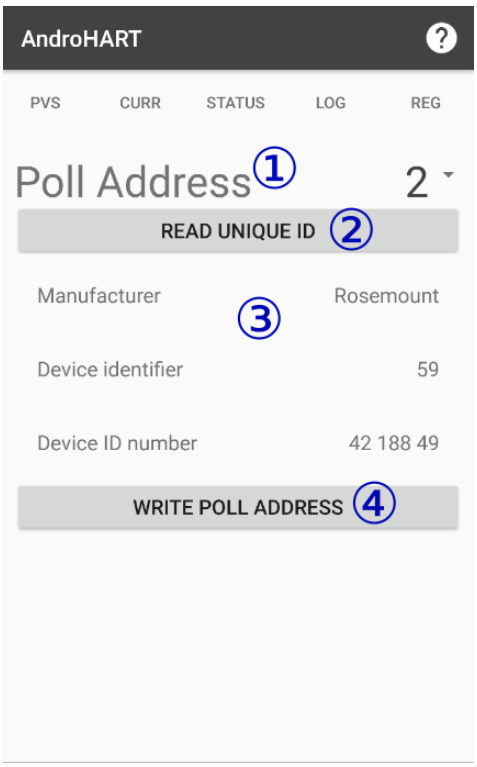
By clicking the question mark icon on the header, the about menu pops up. This contains information, such as the version and a small description.

3.2. Variants

AndroHART has two different versions available on Google Play. The Lite version can not write data back into the field device, data registering works only at a fixed interval and the

amount of stored data is also limited. The Pro version enables those.

3.3. INIT tab



In order to properly configure the device, the app must read some basic identification information. The first questions should be sent here to the device.

3.3.1. Poll Address

To communicate with the filed device using the short address, you must know its poll address. This will be an integer in the range of 0-15. The app does not have a scanning option, if you do not know the poll address, you must try each.

3.3.2. Read unique ID

As other commands use the long address, you must send this command first. Reads the field device's unique identifier which will be used for constructing the long address. Clicking this button invalidates the corresponding fields, these will be filled with three question marks, until the answer is received. As long as you haven't received answer to this command, you can not send others. (Command 0)

3.3.3. Common properties

The most important properties of the connected device are shown here. All three fields are updated by Command 0. The value of manufacturer is read from a predefined array by index.

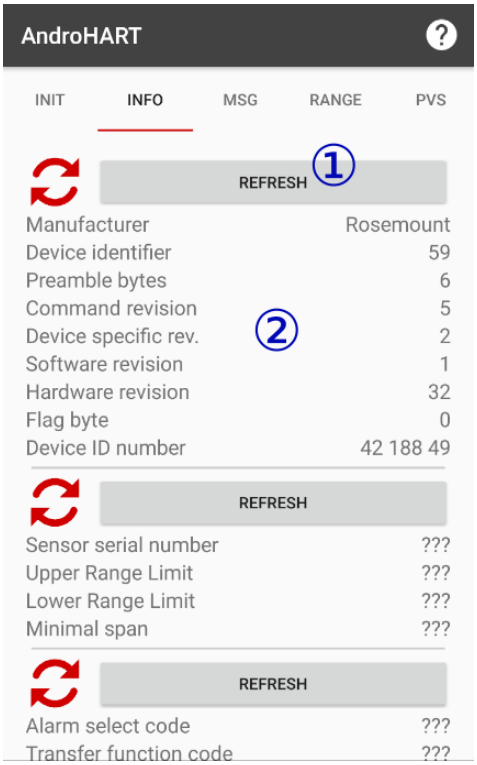
3.3.4. Write poll address

Writes the selected poll address back into the device. The device is called by its long address. It is advised to reread the unique identifier after this command. (Command 6)

Not available in Lite version.

3.4. INFO tab

On this tab, detailed information is displayed about the connected field device.



3.4.1. Refresh

Refreshing values can be done by clicking the Refresh buttons or the red refresh icons. Properties are grouped by command, so clicking a refresh button updates the properties listed below it.

3.4.2. Information block

Displays the the read informations. The format may vary, it can be a single byte, floating point number with a unit, or an element of a list of strings by index. This tab has four information groups.

3.4.2.1. Command 0

Executes the same exact command as ‘Read unique ID’ on the INIT tab. That also means, until the program does not receive answer to this command, you can not send more. Reads in the manufacturer and identifying data. Uses short address.

3.4.2.2. Command 14

Reads in the primary variable related sensor information, such as URL, LRL and minimal span. These are read only properties, you can not change them anywhere in the field device. Uses long address.

3.4.2.3. Command 15

Reads in output informations such as URV, LRV, and damping value. Uses long address.

3.4.2.4. Command 16

Reads in the final assembly number. Uses short address.

3.5. MSG tab

In HART field devices, you have control over three textual fields, which can contain limited length strings.

3.5.1. Message field

An editable field which can handle up to 32 characters. Can accept non ASCII characters, these are handled by the app, since the set of allowed chars in the device is limited to 64 values. You can read below about more specific stuff character encoding.

3.5.2. Message read, write

The ‘Read msg’ button reads in the message stored in the field device, and overwrites the message field with it. The ‘Write msg’ takes the current value in the msg field and writes it back to the field device. Read has a command number of 12, write’s command number is 17.

Write is not available in Lite version.

3.5.3. Tag and description fields

These are editable text fields similar to message. Tag has a max length of 8 characters, while description is limited to 16 characters. Any shorter string is padded with spaces. Read and write commands below handle tag, description and date at the same time.

3.5.4. Date picker

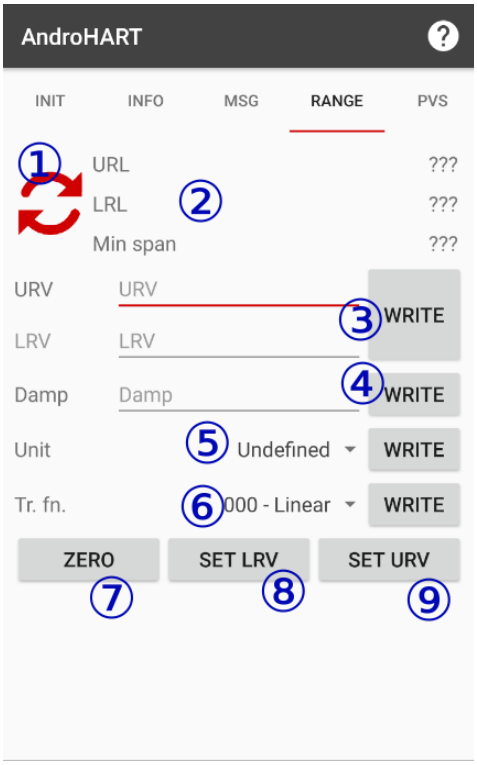
Uses Android’s built in date picker menu, which can be activated by touching the calendar icon. If no date is present, the default shown date is the current. With this method, only correct date values can be given, however nothing guarantees that the field device does not contain invalid values. Day, month, and year values are stored in one byte respectively, so it is possible to store 255th month and so on. Year values are counted from 1900, so 117 (as in byte value) means 2017.

3.5.5. Tag, description, date read and write

The ‘Read’ command overwrites the displayed tag, desc., and date with the read in values, while ‘Write’ does it’s exact opposite. These three fields are treated by the same command. Before executing the write, a confirmation dialog is displayed, where the user can cancel the operation. Read has a command number of 13, write’s command number is 18.

Write is not available in Lite version.

3.6. RANGE tab



3.6.1. Refresh limits

Refreshes the current limit values (URL, LRL, min. span) by reading them in from the field device. Executes command 14, which can be executed on the INFO tab also. The real value of the limits can not be changed, however its numeric value may vary, depending on the selected unit.

3.6.2. Display limits

Shows the current limit values. Its content is basically the same as the second block of the INFO tab. Executing command 14 anywhere updates is.

3.6.3. Range values

These fields are both readable and writable. You can update the values of URV and LRV on the INFO tab (Command 15). You can enter floating point numbers, which you can write into the device by clicking the ‘Write’ button. This handles both values with the same command, so make sure that you enter a correct value into both fields. Write has a command number of 35.

Not available in Lite version, this property is read-only.

3.6.4. Damp

Writable and readable field, into which, you can enter the desired damping value in seconds. Can be updated at the INFO tab. Write has a command number of 34.

Not available in Lite version, this property is read-only.

3.6.5. Unit code

Displays the primary variables’ (URV, LRV, URL, LRL) current unit. Since this is a dropdown list, you can also set another unit. The field device checks the validity of the given unit. Writing the unit back into the device does not update the related fields, however, their numerical value may change. The user gets a warning about this. Command 44.

Not available in Lite version, this property is read-only.

3.6.6. Transfer function

The connection between the measured physical property and the output can be linear, squared, etc. You can select the transfer function from a predefined list. Nothing guarantees that every function is implemented in the field device. After executing the write command, the STATE tab might give information about whether the write was successful.

Command 47.

Not available in Lite version, this property is read-only.

3.6.7. Zero

Nullifies the current measured value, which allows fine-tuning. Not every device can be nullified, the check is done by the field device. (Command 43)

Not available in Lite version.

3.6.8. Set LRV

Sets the current measured primary variable to LRV. (Command 36)

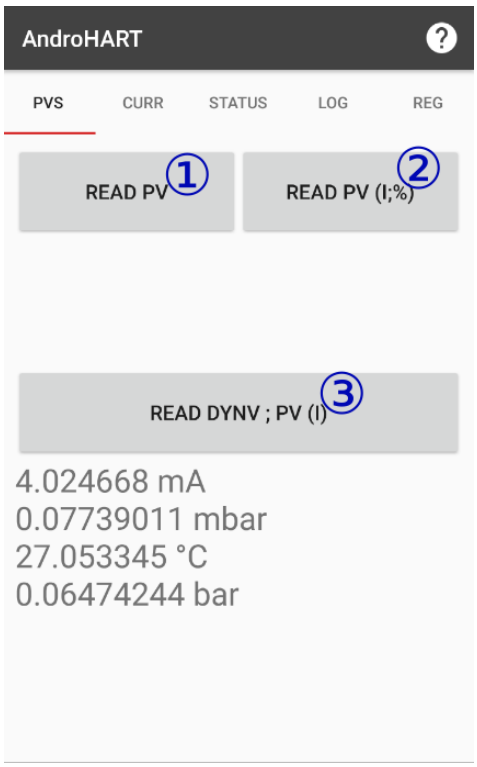
Not available in Lite version.

3.6.9. Set URV

Sets the current measured primary variable to URV. (Command 36)

Not available in Lite version.

3.7. PVS tab



3.7.1. Read PV

Reads in the primary variable (PV) and its unit code, and displays them in a read-only text field. (Command 1)

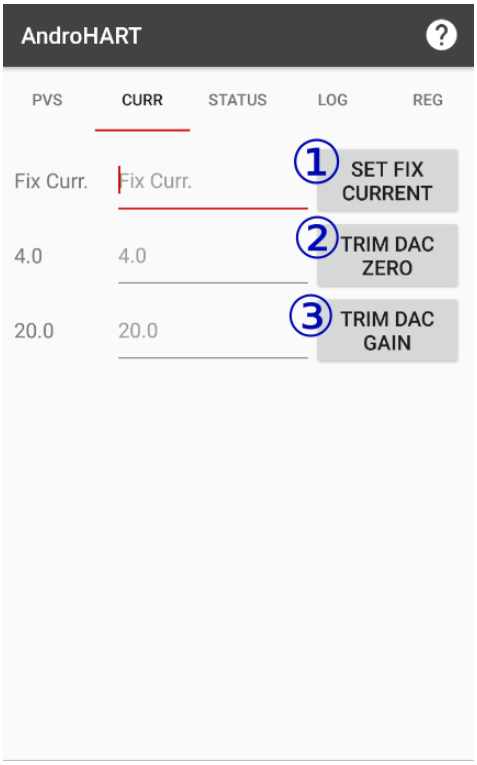
3.7.2. Read PV(I;%)

Reads in the value of current, the percentage value of the current, and displays those in a read-only text field. (Command 2)

3.7.3. Current and dynamic variables

Reads in the value of current (without its unit), and the four predefined dynamic variables (with their units), and displays them in a read-only text field. (Command 3)

3.8. CURR tab



3.8.1. Set fix current

Sets a stable, fixed current in the field device. The application accepts values which does not fit in the 0/4-20mA interval, but the field device will throw an error then. It is not tested, what happens, when you send this command to a device which has a non 0 poll address. In order to turn off the fixed mode, send this command with a parameter of 0, or power off the device. (Command 40)

Not available in Lite version.

3.8.2. Trim dac zero

Only for experienced users. If the current if fixed to 4mA, however, due to the inaccuracy of the device, the real, measured value of the current is not 4mA, you can do some correction. If you input the measured value, and send the command, the field device will do the required correction, and will output a more precise 4mA current. (Command 45)

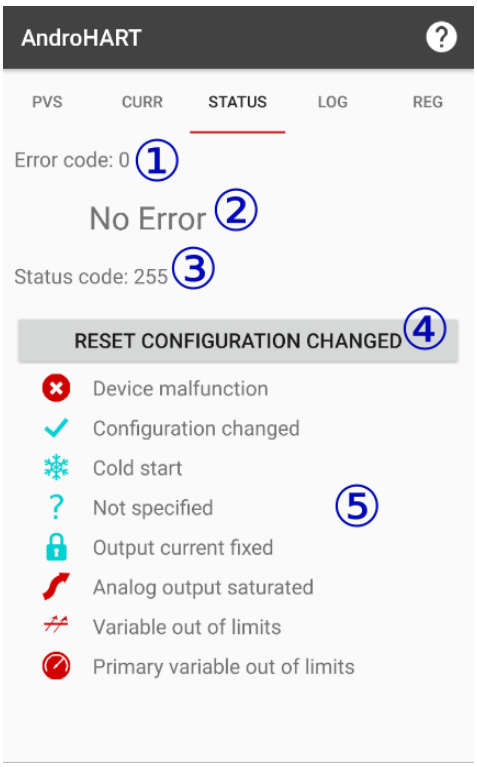
Not available in Lite version.

3.8.3. Trim dac gain

Only for experienced users. If the current if fixed to 20mA, however, due to the inaccuracy of the device, the real, measured value of the current is not 20mA, you can do some correction. If you input the measured value, and send the command, the field device will do the required correction, and will output a more precise 20mA current. (Command 46)

Not available in Lite version.

3.9. STATUS tab



3.9.1. Error code

Displays the error code sent back with the latest response from the device. Error code is stored in one byte, and different codes mean different errors, which are not related to the error code's bits.

3.9.2. Error message

Displays the error message related to the error code. If the sent back error code is not specified in the most common error codes (for example on device specific errors), then 'Device specific error' will be displayed. In order to get more information about these errors, you should contact the provider of the HART device.

3.9.3. Status code

In every response from the device, one byte tells us about the status of the device. This value is bitmasked, each bit means belongs to different status flags.

3.9.4. Reset configuration changed

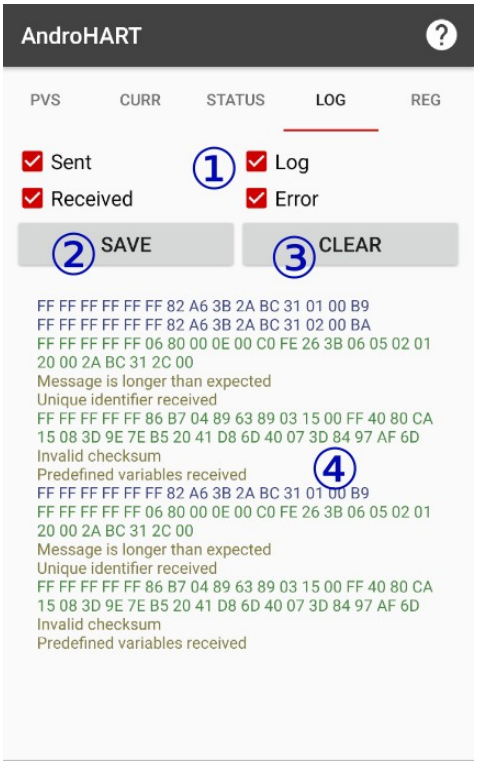
If you change at least one property in the field device, this will be indicated with a status flag. This flag does not count the additional changes and does not reset itself, even when the latest command did not actually change the configuration. To clear this flag, a specific command exists, which can be given by clicking this button. (Command 38)

Not available in Lite version.

3.9.5. Status flags

Displays the different statuses. Statuses with blue icons mean non-error statuses, while red icons mean malfunction and device fault. Next to the icons, the status flags' descriptions are shown.

3.10. LOG tab



3.10.1. Settings

Sent, or received messages and other statuses are tracked in AndroHART. With these four checkboxes, you can set, which types of statuses should be shown in the log. If you uncheck a category, the corresponding events will be still logged, but they will not be displayed. There are four levels of logging:

3.10.1.1. Sent

The sent bytes from the smartphone to the field device in hexadecimal format, separated by spaces. The displayed color is blue.

3.10.1.2. Received

The received bytes from the field device in hexadecimal format separated by spaces. The displayed color is green.

3.10.1.3. Log

The meaning of the received data in a readable format, and non-critical warnings. The displayed color is yellow.

3.10.1.4. Error

Critical errors, which makes the parsing of the received data impossible. The displayed color is red.

3.10.2. Save

By touching the save button, the current logs will be saved in a file, whose name is generated with a timestamp. The saved file's content will be the displayed log messages, therefore the unchecked categories will not be saved. If the file would be empty, the program will not create an empty file. The file will be saved with a .log extension to the application's directory, usually at Android/data/com.kovlev.androhart/files.

Not available in Lite version.

3.10.3. Clear

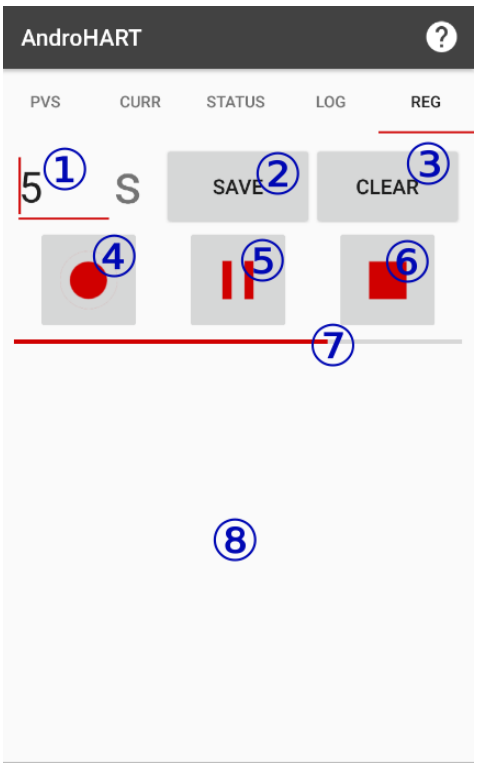
By touching the clear button all logs will be deleted after a confirmation dialog.

3.10.4. The displayed log

Log messages are stored in a textbox (TextView). Since the content of this textbox changes on every checkbox activation, these messages are also stored in the background. To make sure that the program does not run out of memory due to the number of log

messages, sometimes the oldest messages are deleted. The deletion does not care about categories, only the creation time of the messages is taken into consideration. Deletion is a slow process, in an optimal case it happens rarely. At the moment, deletion spares the newest 2000 messages. After the 2000th message a deletion event is possible, after the 3000th message, a deletion event is certain.

3.11. REG tab



AndroHART can be used for data registration with its implemented HART commands. On this tab you can call command 3 periodically and log the responses.

3.11.1. Interval

You can set the time in seconds between data request. The minimal value is 5s, the maximal is 3600s (1 hour). You must restart the registration, to apply the changed interval.

Not available in Lite version, 5s is the fixed interval.

3.11.2. Save

Saves the current registered data into a .tsv file with a timestamped filename. Before saving, a confirmation dialog pops up, which shows the resulting filename. If there is no data, an empty file won't be created. The file will be saved to the app's data directory possibly available at Android/data/com.kovlev.androhart/files.

3.11.3. Clear

Deletes the registered data after a confirmation dialog, even if it is not saved.

3.11.4. Rec

Starts collecting the data if it is possible (if the unique id has already been read). After starting the registration, you can not send other command (the app warns you about this). There are two commands for exception. One is rereading the unique id, which stops the data registering. The other is command 3, which can be executed directly on the PVS tab. This results in an additional data value.

3.11.5. Pause

Pauses the current active registration or continues the current paused registration. On continue, the stopped timer's value is preserved.

3.11.6. Stop

Stops the current registration. The difference between pause and stop is that on stop, the timer value is reset to zero.

3.11.7. Timer

This thin progressbar shows the user the state of the loop. Basically, it is a visual indicator that illustrates the difference between stop and pause.

3.11.8. Registered data

Displays the data values separated with tabulators. To the beginning of each line, a timestamp (date and time in a field) is appended. Also stores the data, the content of the resulting file on save will be exactly the text field's content. Units are also separated with a tab from the corresponding values for easier parsing. No formatting is done here, one record may break up into multiple lines, but these break will not affect the resulting file.

In Lite version this field can only store up to 20 values.

4. Other

4.1. Used libraries

4.1.1. UsbSerial 4.5

Link: https://github.com/felHR85/UsbSerial, license: MIT. This is the underlying driver for serial port, and the application were built on its ‘example’ project. Supported chipsets: CP210X, CDC, FTDI, PL2303, CH34x, CP2130 SPI-USB.

4.1.2. Apache Commons

License: Apache. Helper functions for Java are used.

4.2. Special datatypes

4.2.1.4 byte floating point number

Floating point values are sent on 4 bytes in big endian mode, and is basically the same as the float type in Java. (Standard 4 byte floating point, IEEE 754.)

4.2.2. String encoded on 6 bytes

Textual fields such as message, tag and description are encoded and compressed with 6-bit characters. This means that for example a 32 character strings takes up 24 bytes. This encoding is similar to base64, but slightly differs from it in terms of enabled characters. In this case, enabled characters are all uppercase letters from the English alphabet, digits, and almost every ASCII special non-printable characters. String handling in AndroHART takes multiple steps:

1. Normalizes the string to NFD mode. What this means is that for example a character “é” will be represented on two characters: a letter “e” and a special accent character.
2. Removes these special accent characters.
3. Every remaining non ASCII character is changed to an underscore.
4. Every remaining non-ASCII character is changed to an underscore.
5. Converts the string into uppercase.
6. Every remaining special ASCII character, which are not in the set of allowed characters ({, |, }, `, ~) are changed to an underscore.

With these steps, there should not remain any illegal characters, but if any of those remains, it will not cause problems.

5. Versions

1.0.2.: Adding log system, planning UI.

1.0.4.: Basics of communication.

1.0.5.: Message and tag reading.

1.0.6.: Error handling, poll address, PV read.

1.0.7.: Message and tag reading, data registering.

1.1.3.: First working version, redesigns.

1.1.4.: Fixing bugs.

1.1.6.: Fixing bugs, using custom driver.

1.1.7.: Supporting Android 4.2.

1.1.8.: Google Play ready version.