
BeagleSystems Documentation

BeagleSystems GmbH

Nov 19, 2021

CONTENTS:

1	HowTos	1
1.1	Install the BeagleSystems Software on a drone	1
1.2	Install FT4232H	2
1.3	Compile the documentation	2
1.3.1	To generate a pdf using latex	2
1.4	Debug PX4 topics	3
1.5	How to debug USB	4
2	Hardware	5
2.1	Payloads	5
2.1.1	EH2000	5
3	Reference	7
3.1	ROS Messages	8
3.1.1	ROS message definitions	8
3.2	MQTT interface	9
3.2.1	MQTT interface configuration	10
3.3	mqtt_bridge	10
3.3.1	mqtt_bridge package	10
4	Precision Landing	13
4.1	Precision Landing State Machine	13
4.2	Precision Landing Integration with Gimbal Camera	13
	Python Module Index	15
	Index	17

HOWTOS

1.1 Install the BeagleSystems Software on a drone

To gain access to the repository, we first have to do some setup:

```
$ git config --global http.postBuffer 524288000
$ ssh-keygen -t rsa -b 4096 -C "your_email@beaglesystems.com"
$ cat ~/.ssh/id_rsa.pub
```

Upload the public key to your github settings to be able to download the repository.

```
$ mkdir Development
$ cd Development
$ git clone git@github.com:BeagleSystems/BeagleComrade -b develop
```

Create the file `~/beaglerc` with the following content.

```
#!/bin/bash

# !!! Double check that these paths are correctly set !!!
source /opt/ros/noetic/setup.bash
source /home/david/Development/beaglesystems/BeagleComrade/devel/setup.bash
export ROS_PACKAGE_PATH=/home/david/Development/beaglesystems/PX4-Autopilot:/home/
↪david/Development/beaglesystems/PX4-Autopilot/Tools/sitl_gazebo:$ROS_PACKAGE_PATH

# These exports are necessary if you try to run ROS debugging tools connected via_
↪WiFi to the drone. Use the same exports on your local computer
#export ROS_HOSTNAME=192.168.8.100
#export ROS_IP=$ROS_HOSTNAME
#export ROS_MASTER_URI=http://$ROS_HOSTNAME:11311

export GST_PLUGIN_PATH=/usr/local/lib/aarch64-linux-gnu/gstreamer-1.0:/usr/lib/
↪aarch64-linux-gnu/gstreamer-1.0
export LD_LIBRARY_PATH=/usr/local/lib/aarch64-linux-gnu:$LD_LIBRARY_PATH
```

1.2 Install FT4232H

```
..code-block:: sh
    sudo apt-get install libconfuse-dev
```

1.3 Compile the documentation

We adhere to the recommendations described on <https://www.writethedocs.org/guide/> and <https://documentation.divio.com/reference/>. Reference guides are kept in a similar style as <http://mavlink.io/en/services/mission.html>.

```
sudo pip3 install sphinxcontrib-mermaid
sudo pip3 install sphinx-jinja sphinxcontrib-napoleon sphinx-rtd-theme
```

Install sphinx-bootstrap-theme:

```
cd ~/Development/beaglesystems
git clone git@github.com:dayjaby/sphinx-bootstrap-theme
cd sphinx-bootstrap-theme
sudo python3 setup.py install
```

Modify the CSS:

```
cd ~/Development/beaglesystems
git clone git@github.com:dayjaby/bootstrap
cd bootstrap
npm install
npm install grunt
... modify files in ./less/ ...
./node_modules/.bin/grunt dist
cp -r dist/* ~/Development/beaglesystems/sphinx-bootstrap-theme/bootstrap/static/
↪bootstrap-3.4.1/
```

Create the documentation:

```
sphinx-apidoc -f -o source/mqtt_bridge ../BeagleComrade/src/mqtt_bridge/src/mqtt_
↪bridge
make clean && make html
```

Instead of 'make html' you can create the documentation via

```
python3 -msphinx . _build
```

1.3.1 To generate a pdf using latex

```
sudo apt install texlive-full
sudo apt install texlive-latex-extra
make latexpdf
```

The pdf file is located in doc/_build/latex/

1.4 Debug PX4 topics

NuttX shell, list all available commands and list all uORB topics:

```
nsh> help
nsh> uorb top
q
```

If you have GPS problems, please check:

```
nsh> listener vehicle_gps_position
```

Is jamming indicator below 40? If not, make sure that all USB3.0 cables are far away from the GPS sensor.

In a SITL environment, we can use GDB to analyze. I assume that you run PX4 via a robot_upstarted launch file. Make sure that the PX4 ros node is commented out. First, we have to compile PX4 in GDB mode:

```
DONT_RUN=1 make px4_sitl_default gazebo___gdb
```

Next, we start the node as root.

```
$ sudo su -
# export HOME=/home/beagle
# source $HOME/.beaglerc
# gdb --args $HOME/Development/beaglesystems/PX4-Autopilot/build/px4_sitl_rtps/bin/
→ px4 /$HOME/Development/beaglesystems/PX4-Autopilot/ROMFS/px4fmu_common -s etc/init.
→ d-posix/rcS -t /$HOME/Development/beaglesystems/PX4-Autopilot/test_data
(gdb) handle SIGCONT noprint nostop
(gdb) run
```

One example how to break on certain mavlink messages, e.g. with mavlink message ID 212:

```
<ctrl-c> if GDB is running already
(gdb) break MavlinkReceiver::handle_message
(gdb) continue
Thread 75 "mavlink_rcv_if1" hit Breakpoint 1, MavlinkReceiver::handle_message_
→ (this=0x7fff8c000b20, msg=0x7fffc6ffad90)
at ../../src/modules/mavlink/mavlink_receiver.cpp:132
132      {
(gdb) x/20i $pc
=> 0x55555561d420 <MavlinkReceiver::handle_message(__mavlink_message*)>:      push    _
→ %rbp
    0x55555561d421 <MavlinkReceiver::handle_message(__mavlink_message*)+1>:      push    _
→ %rbx
    0x55555561d422 <MavlinkReceiver::handle_message(__mavlink_message*)+2>:      mov     _
→ %rsi,%rbx
    0x55555561d425 <MavlinkReceiver::handle_message(__mavlink_message*)+5>:      mov     _
→ %rdi,%rbp
    0x55555561d428 <MavlinkReceiver::handle_message(__mavlink_message*)+8>:      sub     _
→ $0x8,%rsp
    0x55555561d42c <MavlinkReceiver::handle_message(__mavlink_message*)+12>:      _
→ movzbl 0xa(%rbx),%eax
    0x55555561d430 <MavlinkReceiver::handle_message(__mavlink_message*)+16>:      _
→ movzbl 0x9(%rsi),%esi
    0x55555561d434 <MavlinkReceiver::handle_message(__mavlink_message*)+20>:      _
→ movzbl 0xb(%rbx),%edx
    0x55555561d438 <MavlinkReceiver::handle_message(__mavlink_message*)+24>:      shl     _
→ $0x8,%rax
```

(continues on next page)

(continued from previous page)

```

0x55555561d43c <MavlinkReceiver::handle_message(__mavlink_message*)+28>:    or    ❷
↳ %rsi,%rax
0x55555561d43f <MavlinkReceiver::handle_message(__mavlink_message*)+31>:    shl    ❷
↳ $0x10,%rdx
0x55555561d443 <MavlinkReceiver::handle_message(__mavlink_message*)+35>:    or    ❷
↳ %rdx,%rax
0x55555561d446 <MavlinkReceiver::handle_message(__mavlink_message*)+38>:    cmp    ❷
↳ $0x8a,%eax
0x55555561d44b <MavlinkReceiver::handle_message(__mavlink_message*)+43>:    je     ❷
↳ 0x55555561d8d0 <MavlinkReceiver::handle_message(__mavlink_message*)+1200>
(gdb) delete
(gdb) break *0x55555561d446 if $eax == 212
(gdb) info registers $eax
(gdb) continue

```

1.5 How to debug USB

```

sudo apt install libboost-dev libpcap-dev
git clone https://github.com/aguinete/usbtop
cd usbtop
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make
sudo make install
sudo modprobe usbmon
sudo usbtop

```

Compare that with the USB devices found via `lsusb`.

USB devices can fail if the voltage drops below 5V. To check the voltage on Jetson Nano, run:

```
cat /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in_voltage0_input
```


HARDWARE

2.1 Payloads

2.1.1 EH2000

Test commands for the NuttX shell

Get the device information and reported feedback from the gimbal:

```
listener gimbal_device_information  
listener gimbal_device_attitude_status
```

Do a camera trigger test:

```
camera_trigger test
```

Do a continuous camera trigger test until stopped:

```
camera_trigger test_interval  
camera_trigger test_interval stop
```

Make the camera look forward and follow yaw:

```
eh2000 test follow
```

Make the camera look down and follow yaw:

```
eh2000 test lookdown
```

Make the camera look left/right with a pitch of 45 degrees down:

```
eh2000 test lookleft  
eh2000 test lookright
```

We provide commands to test the camera zoomed in (50mm) and zoomed out (16mm) and automatically focussed:

```
eh2000 test zoomin  
eh2000 test zoomout  
eh2000 test focus
```

Prepare the camera for precision landing, which includes the following commands:

- zoom out (MAV_CMD_SET_CAMERA_ZOOM)

- auto focus (MAV_CMD_SET_CAMERA_FOCUS)
- follow yaw (MAV_CMD_DO_GIMBAL_MANAGER_PITCHYAW)
- lockdown (MAV_CMD_DO_GIMBAL_MANAGER_PITCHYAW)

```
eh2000 test precland
```

We prepared some profiles for the camera:

```
eh2000 test profile_auto
eh2000 test profile_shutter
```

Be aware that these commands do certain other things: They flash the SD card and set the save path, so that images are written to the SD card.

As a fallback option, ssh to the drone and run these commands:

```
# Format the SD card
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=formatMedia"
# Switch to manual mode
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=shootMode&mode=5"
# Set aperture to F5.6
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=apertureMode&
↪mode=16"
# Set ISO mode to AUTO
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=isoMode&mode=0"
# Set shutter speed to 1/2500
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=shutterSpeedMode&
↪mode=18"
# Set exposure compensation to -0.3EV
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?
↪action=exposureCompensationMode&mode=4"
# Set zoom to 0 (completely zoomed out)
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=setZoomValue&
↪value=0"
# Save images to SD card
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=setSavePath&path=1"
# Do a single capture
curl -G "http://192.168.42.108:80/cgi-bin/configManager.cgi?action=capture&mode=0"
```

CHAPTER
THREE

REFERENCE

3.1 ROS Messages

3.1.1 ROS message definitions

Table 1: Mapping of MQTT and ROS topics

ROS message	Data format
beagle_interfaces/RtcmData	string house_id string rtc_m_id uint32 length string data

3.2 MQTT interface

Connecting to MQTT

The MQTT broker is hosted on 40.87.148.13:1883.

Table 2: Mapping of MQTT and ROS topics

MQTT topic	In/Out	ROS topic	Data format	Timestamp ¹
house+/rtcm+/raw	→	/rtk/rtcm	<i>beagle_interfaces/RtcmData</i>	

¹ The timestamp is an additional field in a message and is based on `time.time()` at the time of transmission of the data. Timestamps of the samples themselves are not included if not stated otherwise.

3.2.1 MQTT interface configuration

```
mqtt:
  client:
    protocol: 4          # MQTTv311
    client_id_from_mac: ["eth2", "enol", "eth1", "eth0"]
  connection:
    host: "40.87.148.13"
    port: 1883
    keepalive: 10
  account:
    username: "beagle"
    password: "beagleB0o12"
  will:
    topic: ~/disconnected
    payload: "{}"
    qos: 2
  disconnect_on_shutdown: False
serializer: json:dumps
deserializer: json:loads
bridge:
  - factory: mqtt_bridge.bridge:MqttToRosBridge
    msg_type: beagle_interfaces.msg.RtcData
    topic_from: house/+/rtcm/+/raw
    topic_to: /rtk/rtcm
    wildcards: ["house_id", "rtcm_id"]
```

3.3 mqtt_bridge

3.3.1 mqtt_bridge package

Submodules

mqtt_bridge.app module

```
mqtt_bridge.app.mqtt_bridge_node()
```

mqtt_bridge.bridge module

```
class mqtt_bridge.bridge.Bridge
```

Bases: object

Bridge base class

Parameters

- **_mqtt_client** (*mqtt.Client*) – MQTT client
- **_serialize** – message serialize callable
- **_deserialize** – message deserialize callable

```
static is_service()
```

```
class mqtt_bridge.bridge.MqttToRosBridge (topic_from, topic_to, msg_type, frequency=None,  
                                           qos=2, queue_size=10, wildcards=None,  
                                           latch=False)
```

Bases: `mqtt_bridge.bridge.Bridge`

Bridge from MQTT to ROS topic

Parameters

- **topic_from** (*str*) – incoming MQTT topic path
- **topic_to** (*str*) – outgoing ROS topic path
- **msg_type** (*class*) – subclass of ROS Message
- **frequency** (*float|None*) – publish frequency
- **queue_size** (*int*) – ROS publisher’s queue size
- **wildcards** (*list-of-str|None*) – list of wildcards. If it is not None, replace any + in topic_from with the values in this list.
- **latch** (*bool|False*) – whether to latch the message

```
class mqtt_bridge.bridge.RosToMqttBridge (topic_from, topic_to, msg_type, fre-  
                                           quency=None, qos=2, retain=False,  
                                           delete_retained_on_shutdown=False, wild-  
                                           cards=None, drop=None, timestamp=False,  
                                           bool_convert=None)
```

Bases: `mqtt_bridge.bridge.Bridge`

Bridge from ROS topic to MQTT

Parameters

- **topic_from** (*str*) – incoming ROS topic path
- **topic_to** (*str*) – outgoing MQTT topic path
- **msg_type** (*class*) – subclass of ROS Message
- **frequency** (*float|None*) – publish frequency
- **qos** (*int|2*) – MQTT QoS
- **retain** (*bool|False*) – whether to retain the message
- **delete_retained_on_shutdown** (*bool|False*) – delete the message on shutdown if it was retained
- **drop** (*list-of-str|None*) – if it is not None, delete all values for the given keys

```
class mqtt_bridge.bridge.RosToMqttServiceBridge (topic, msg_type, qos=2)
```

Bases: `mqtt_bridge.bridge.Bridge`

Bridge from ROS topic to MQTT

Parameters

- **topic** (*str*) – incoming ROS topic path
- **msg_type** (*class*) – subclass of ROS Service
- **qos** (*int|2*) – MQTT QoS

```
static is_service ()
```

`mqtt_bridge.bridge.create_bridge` (*factory*, *msg_type*, ***kwargs*)
bridge generator function

Parameters

- **factory** (*str/class*) – Bridge class
- **msg_type** (*str/class*) – ROS message type
- **kwargs** (*dict*) – a dictionary of arguments for the bridge class initialization

Return `Bridge` bridge object

mqtt_bridge.mqtt_client module

`mqtt_bridge.mqtt_client.create_private_path_extractor` (*mqtt_private_path*)

`mqtt_bridge.mqtt_client.default_mqtt_client_factory` (*params*, *private_path_extractor*) *pri-*

MQTT Client factory

Parameters **params** (*dict*) – configuration parameters

Return `mqtt.Client` MQTT Client

mqtt_bridge.util module

`mqtt_bridge.util.extract_values` (*inst*)

`mqtt_bridge.util.lookup_object` (*object_path*, *package='mqtt_bridge'*)
lookup object from a `some.module:object_name` specification.

`mqtt_bridge.util.populate_instance` (*msg*, *inst*)

Returns an instance of the provided class, with its fields populated according to the values in `msg`

Module contents

PRECISION LANDING

4.1 Precision Landing State Machine

4.2 Precision Landing Integration with Gimbal Camera

PYTHON MODULE INDEX

m

- `mqtt_bridge`, [12](#)
- `mqtt_bridge.app`, [10](#)
- `mqtt_bridge.bridge`, [10](#)
- `mqtt_bridge.mqtt_client`, [12](#)
- `mqtt_bridge.util`, [12](#)

INDEX

B

Bridge (class in *mqtt_bridge.bridge*), 10

C

create_bridge() (in module *mqtt_bridge.bridge*), 11

create_private_path_extractor() (in module *mqtt_bridge.mqtt_client*), 12

D

default_mqtt_client_factory() (in module *mqtt_bridge.mqtt_client*), 12

E

extract_values() (in module *mqtt_bridge.util*), 12

I

is_service() (*mqtt_bridge.bridge.Bridge* static method), 10

is_service() (*mqtt_bridge.bridge.RosToMqttServiceBridge* static method), 11

L

lookup_object() (in module *mqtt_bridge.util*), 12

M

module

mqtt_bridge, 12

mqtt_bridge.app, 10

mqtt_bridge.bridge, 10

mqtt_bridge.mqtt_client, 12

mqtt_bridge.util, 12

mqtt_bridge

 module, 12

mqtt_bridge.app

 module, 10

mqtt_bridge.bridge

 module, 10

mqtt_bridge.mqtt_client

 module, 12

mqtt_bridge.util

 module, 12

mqtt_bridge_node() (in module *mqtt_bridge.app*), 10

MqttToRosBridge (class in *mqtt_bridge.bridge*), 10

P

populate_instance() (in module *mqtt_bridge.util*), 12

R

RosToMqttBridge (class in *mqtt_bridge.bridge*), 11

RosToMqttServiceBridge (class in *mqtt_bridge.bridge*), 11