GitHub This repository Search Explore Features Enterprise Pricing Sign up Sign in

dimasad / **pdva-pilot**

Watch 2 Star 0 Fork 0

Branch: **master** pdva-pilot / **README.md**

RafaelTupynamba on 27 Mar Updated README: change in use of mavlink python library

1 contributor

326 lines (221 sloc) 8.986 kB

Raw Blame History

PDVA pilot

Software for the UAV controller developed at UFMG.

Instructions

Connecting to Gumstix

Possible ways to connect to the Gumstix device are via ethernet, via minicom

http://docwiki.gumstix.org/index.php?title=Connecting_via_Serial_-_Linux#Connecting_with_Minicom

or via serial connection

http://gumstix.org/connect-to-my-gumstix-system.html

To login as root, use password `vantufmg` .

Yocto Project

To start, you should have a Linux partition with at least 60 GB on your computer. Follow the instructions in

https://github.com/gumstix/Gumstix-YoctoProject-Repo

to get started with the Yocto Project build system. This will allow you to compile the image of the operating system. The whole process takes many hours and downloads more than 25 GB. The images are created in **yocto/build/tmp/deploy/images**.

The important files are

```
MLO
u-boot.img
uImage
gumstix-console-image-overo.tar.bz2
```

To transfer the files to the Gumstix device, first you need to prepare the microSD card:

Formatting the microSD card

This site:

http://gumstix.org/create-a-bootable-microsd-card.html

shows how to prepare the microSD card to receive a bootable image of the operating system.

Meta-vant-finep

After the previous step, you should already have a working Linux distribution on the SD card. Test the system boot (for example, connecting via minicom). Now, you should just compile the image again, including the meta-vant-finep layer (it won't take as long as the previous compilation).

Add the layer

https://github.com/dimasad/meta-vant-finep

to your repo and include in the file **yocto/.repo/manifests/default.xml** the following lines

```
<remote name="dimasad" fetch="git://github.com/dimasad" />
<project name="meta-vant-finep" path="poky/meta-vant-finep" remote="dimasad" revision="master" upstream="
```

Run `repo sync` at the yocto folder and in the file **yocto/build/conf/bblayers.conf** add a line

```
${OEROOT}/meta-vant-finep \
```

under the variable `BBLAYERS`. Then you will be ready to compile the `gumstix-console-image` again.

Check that the SPI device is created at **/dev/spidev1.1**.

One important part of the meta-vant-finep layer is the recipe in

https://github.com/dimasad/meta-vant-finep/tree/master/recipes-pdva

This recipe is used to cross compile the pdva-pilot project and generate an executable which can run on the Gumstix device.

Install libconfig and glib

Install the package libglib2.0-dev using apt-get and the package libconfig-dev according to the following instructions.

```
cd /tmp
wget http://www.hyperrealm.com/libconfig/libconfig-1.4.9.tar.gz
tar -xf libconfig-1.4.9.tar.gz
cd libconfig-1.4.9
./configure
make
sudo make install
```

Install MAVLink

To install the MAVLink library

```
cd /tmp/
git clone https://github.com/mavlink/mavlink.git
mkdir generated
wget https://raw.github.com/dimasad/pdva-pilot/master/pdvapilot.xml -O mavlink/message_definitions/v1.0/p
export PYTHONPATH=$PYTHONPATH:/tmp/mavlink
python -m pymavlink.tools.mavgen --output=generated --lang=C mavlink/message_definitions/v1.0/pdvapilot.x
sudo rm -r /usr/local/include/mavlink
sudo mkdir /usr/local/include/mavlink
sudo mv generated /usr/local/include/mavlink/v1.0
```

The message fields are described in

https://github.com/dimasad/pdva-pilot/blob/master/pdvapilot.xml

Note that the sensor head runs on a DSP that has 16-bit bytes. This causes a lot o trouble to the MAVLink messages, so the MAVLink code on the sensor head had to be rewritten to take this into account. All the fields in the messages sent to and from the sensor head should have sizes multiple of 16 bits (do not use unit8_t or int8_t).

This website gives an example of how to use MAVLink to send and receive messages:

http://qgroundcontrol.org/dev/mavlink_onboard_integration_tutorial

Compiling the PDVA pilot code

Download the pdva-pilot code from

https://github.com/dimasad/pdva-pilot

and place the pdva-pilot folder in your project root (the same folder that contains the yocto folder). Then enter the pdva-pilot folder.

To compile the code for your computer, you can run

```
mkdir -p build
cd build/
cmake ..
make
```

Cross compiling the PDVA pilot code

To compile the code to run on the Gumstix, follow these instructions.

For new commits (using bitbake pdva-pilot)

```
git commit
git push
cd ../yocto/
repo sync
source poky/oe-init-build-env
bitbake pdva-pilot -c clean
bitbake pdva-pilot

cd tmp/deploy/rpm/armv7a_vfp_neon

# copy rpm files to Gumstix
sudo cp pdva-pilot-0.0+gitmaster-r0.armv7a_vfp_neon.rpm /media/rootfs_/home/root/rpm/

# To install, run this command on the Gumstix device
zypper install pdva-pilot-0.0+gitmaster-r0.armv7a_vfp_neon.rpm
```

For simple changes (using bitbake -c compile --force)

```
rsync ./* ../yocto/build/tmp/work/armv7a-vfp-neon-poky-linux-gnueabi/pdva-pilot-0.0+gitmaster-r0/pdva-pil
cd ../yocto/
repo sync
source poky/oe-init-build-env
bitbake -b ../poky/meta-vant-finep/recipes-pdva/pdva-pilot/pdva-pilot_git.bb -c compile --force
sudo cp tmp/work/armv7a-vfp-neon-poky-linux-gnueabi/pdva-pilot-0.0+gitmaster-r0/pdva-pilot-0.0+gitmaster/
```

For simple changes (using make)

```
rsync ./* ../yocto/build/tmp/work/armv7a-vfp-neon-poky-linux-gnueabi/pdva-pilot-0.0+gitmaster-r0/pdva-pil
cd ../yocto/build/tmp/work/armv7a-vfp-neon-poky-linux-gnueabi/pdva-pilot-0.0+gitmaster-r0/pdva-pilot-0.0+
make
sudo cp pdva-pilot /media/rootfs_/home/root/binaries/
```

Program structure

Parameters loaded from configuration files

The files **pdva-pilot.cfg** and **mav_params.cfg** located at the directory defined by **PDVA_CONFIG_DIR** are used to store configuration parameters that are loaded at the start of program execution.

The available parameters are

```
uint8_t sysid;
time_t control_timer_period_s;
long control_timer_period_ns;
time_t datalog_timer_period_s;
long datalog_timer_period_ns;

int control_id;

    uint32_t spi_speed_hz;

int datalog_write_ms;

int downsample_%s;
```

(where %s is the name of the datalog file)

```
int filter_%s_n;
```

(where %s is the name of the datalog file)

```
double filter_%s_a_%d;
```

(where %s is the name of the datalog file and %d is the index for the denominator)

```
double filter_%s_b_%d;
```

(where %s is the name of the datalog file and %d is the index for the numerator)

```
acc_%d_gain
acc_%d_offset
gyro_%d_gain
gyro_%d_offset
gyro_temp_gain
gyro_temp_offset
mag_%d_gain
mag_%d_offset
dyn_press_gain
dyn_press_offset
stat_press_gain
stat_press_offset
```

(where %d is the index for the axis 0,1,2)

```
att_est_%d_gain
att_est_%d_offset
airspeed_gain
airspeed_offset
altitude_gain
altitude_offset
```

(where %d is the index for the axis 0,1,2)

```
lat_gps_gain
lat_gps_offset
lon_gps_gain
```

```
lon_gps_offset
alt_gps_gain
alt_gps_offset
hdg_gps_gain
hdg_gps_offset
speed_gps_gain
speed_gps_offset
pos_fix_gps_gain
pos_fix_gps_offset
nosv_gps_gain
nosv_gps_offset
hdop_gps_gain
hdop_gps_offset

aileron_gain
aileron_offset
elevator_gain
elevator_offset
throttle_gain
throttle_offset
rudder_gain
rudder_offset
```

Datalog

The datalog files are created at the directory defined at **DATALOG_DIR**. In this directory, the file **last_experiment** contains the number of the last experiment performed. One folder is created for each experiment with the following files

```
sensor
attitude
gps
control
telecommand
```

Each file has its own downsample factor (the file is written with a period of M*datalog_timer_period, where M is the downsample factor of the file.)

A low-pass digital filter can be implemented for each file. The order of the filter and its coefficients can be set up in the configuration file.

For example, the parameters

```
filter_sensor_n = 1;
filter_sensor_a_0 = 1.0;
filter_sensor_a_1 = -0.1;
filter_sensor_b_0 = 0.9;
filter_sensor_b_1 = 0.0;
```

create a filter for the sensor file with order 1 given by the difference equation y[k] = 0.1y[k-1] + 0.9x[k].

When the variables are not set in the configuration files, they use default values: the datalog_timer_period_ns and the control_timer_period_ns use the value CONTROL_TIMER_PERIOD_NS, all the downsample factors are set to 1 and no low-pass filter is used. More precisely, the filters are set to y[k] = x[k], (transfer function 1).

Threads

The program is divided in threads and it is possible to set the scheduling policy and priority of each thread.

Authors

Project started by Dimas Abreu Dutra and continued by Rafael Tupynamba Dutra.

Universidade Federal de Minas Gerais.