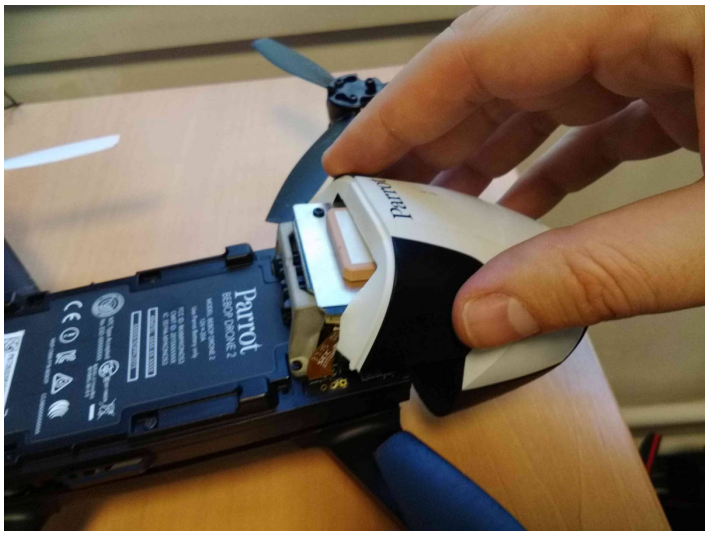
Hacking a Commercial Drone to run an Open Source Autopilot - APM on Parrot Bebop

Julien BERAUD

April 5th 2016 - Embedded Linux Conference San Diego



Parrot

Introduction

Architecture and Porting

Running Ardupilot on a Bebop 2

Optical Flow

Sonar

Monitoring real-time performances with LTTng

Conclusion

Introduction

Parrot Bebop

- 410g
- Parrot P7 SoC (dual Cortex A9)
- IMU, Barometer, Compass, Vertical Camera, Sonar, GPS
- Linux kernel 3.4 (no mainline support)
- Front camera with fish-eye lens

Parrot

Parrot Bebop 2

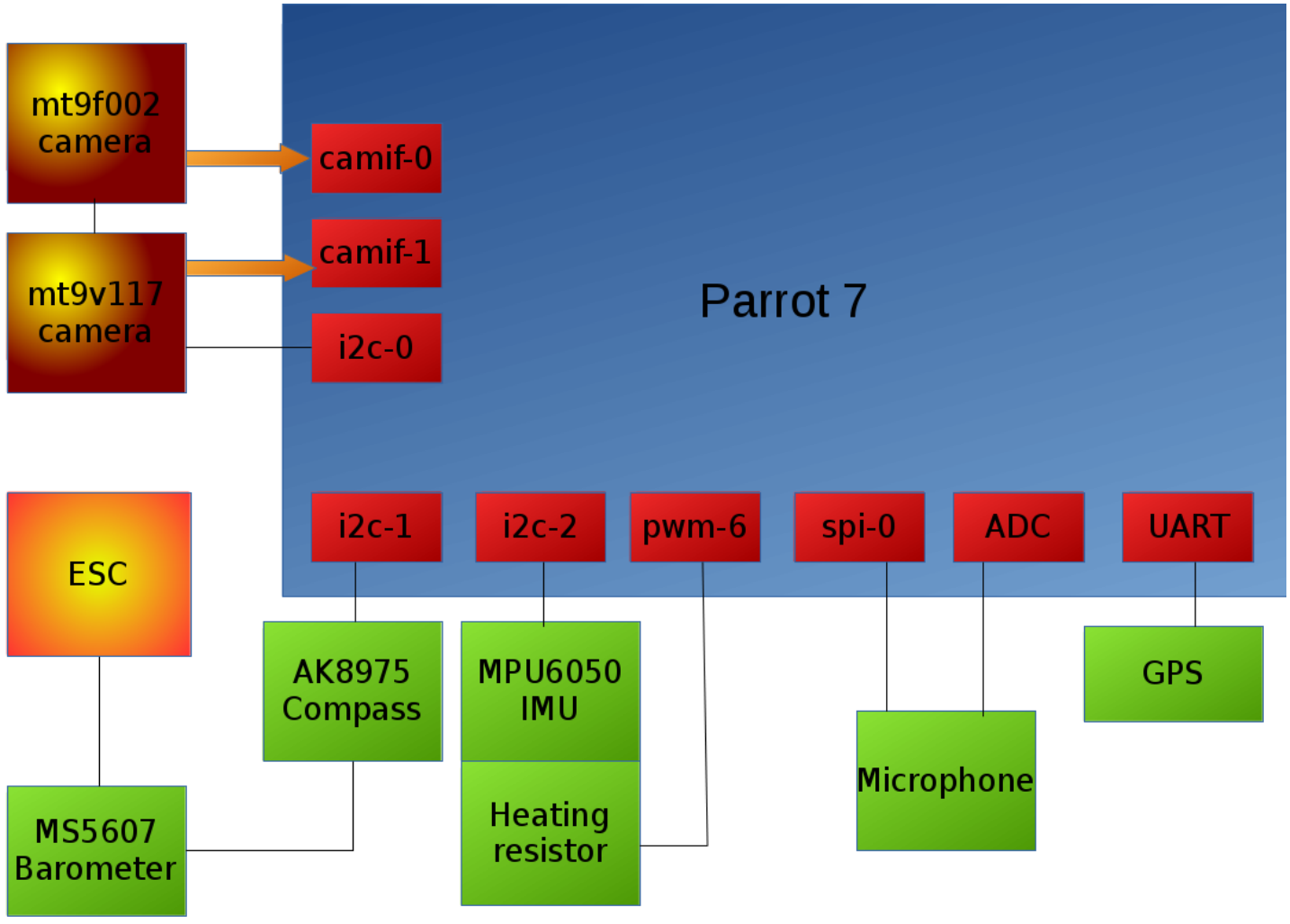
- 500g
- Parrot P7 SoC (dual Cortex A9)
- IMU, Barometer, Compass, Vertical Camera, Sonar, GPS
- Linux kernel 3.4 (no mainline support)
- Front camera with fish-eye lens

What is an autopilot ?

- Quad-Copters are just too difficult for humans to handle
- They need software to be controlled properly
- Autopilot software
  - Inputs : sensors (IMU, Compass, Baro, Sonar, Cameras, ...)
  - Inputs : user commands
  - Outputs : Propeller speeds

Architecture and Porting

Hardware architecture



Parrot

Linux integration

- i2c-dev
- spidev
- UART
- v4l2
- network interface (802.11)
- sysfs pwm/gpio
- iio

Linux integration

```
root@milosboard:/ # ls /dev
IMU                 media0              tty1                tty40               ttyPA1
Magnetometer        mem                 tty10               tty41               uart-0
Motors              mmcblk0             tty11               tty42               uart-1
Pressure            mmcblk0boot0        tty12               tty43               ubi0
android_adb         mmcblk0boot1        tty13               tty44               ubi0_0
bus                 mt9f002             tty14               tty45               ubi1
console             mt9v117             tty15               tty46               ubi1_0
cpu_dma_latency     mtd0                tty16               tty47               ubi2
fb0                 mtd0ro              tty17               tty48               ubi2_0
fd                  mtd1                tty18               tty49               ubi2_1
full                mtd1ro              tty19               tty5                ubi_ctrl
hx280a              mtd2                tty2                tty50               ulog_kmsgd
i2c-0               mtd2ro              tty20               tty51               ulog_main
i2c-1               mtd3                tty21               tty52               ump
i2c-2               mtd3ro              tty22               tty53               urandom
i2c-akm8963         mtd4                tty23               tty54               usb_accessory
i2c-cypress         mtd4ro              tty24               tty55               usbdev1.1
i2c-mpu6050         mtp_usb             tty25               tty56               usbdev1.2
i2c-ms5607          network_latency     tty26               tty57               v4l-subdev0
i2c-mt9f002         network_throughput  tty27               tty58               v4l-subdev1
i2c-mt9v117         null                tty28               tty59               vcs
i2c-p7mu            p7ump               tty29               tty6                vcs1
iio:device0         ptmx                tty3                tty60               vcsa
iio:device1         pts                 tty30               tty61               vcsa1
iio:device2         random              tty31               tty62               video0
iio:device3         rtc0                tty32               tty63               video1
iio:device4         shm                 tty33               tty7                video2
iio:device5         socket              tty34               tty8                video3
iio:device6         spidev1.0           tty35               tty9                video4
kmem                stderr              tty36               ttyGS0              video5
kmsg                stdin               tty37               ttyGS1              zero
loop-control        stdout              tty38               ttyGS2
loop0               tty                 tty39               ttyGS3
mali                tty0                tty4                ttyPA0
root@milosboard:/ #
```

Ardupilot

Ardupilot

Ardupilot (APM)

- Open Source - GPLv3
- Originally developed to run on an Arduino
- C++
- Some linux boards already supported before Bebop

Software architecture

- Vehicle specific flight code (ArduCopter, ArduPlane, ArduRover)
- Shared libraries that include sensor drivers
- Hardware Abstraction Layer providing access to platform-specific methods
- AP_HAL_Linux giving access to spidev, i2c-dev, uart drivers, etc...

Drivers and developments to support Bebop board

Developments needed to add support for Bebop

- MPU6000 driver adaptation for MPU6050 over i2c and FIFO
- AK8963 driver adaptation for direct connection
- MS5611 driver adaptation to support MS5607
- NMEA GPS driver modifications to handle some frames
- Driver for the motor controller (ESC) over i2c
- Remote controller

Inertial Measurement Unit

Inertial Measurement Unit

Inertial Measurement Unit

- Accelerometer and gyroscope
- Gives a 3D acceleration vector (x,y,z)
- Gives a 3D angular speed vector (roll, pitch, yaw)
- MPU6050 runs over i2c
- 8kHz maximum gyros and 1kHz maximum acceleros

MPU6050 (1/3)

- Driver for MPU6000 over spi
- Timer at 1kHz to read datas (1 sample per ms)
- Works over spi with PREEMPT_RT patch
- I2c bus too slow
- No PREEMPT_RT patch on the Bebop
- Some samples are missed

MPU6050 (2/3)

```
void AP_InertialSensor_MPU6000::_read_fifo()
{
    uint8_t n_samples;
    uint16_t bytes_read;
    uint8_t rx[MAX_DATA_READ];

    if (!_block_read(MPUREG_FIFO_COUNTH, rx, 2)) {
        hal.console->printf("MPU60x0: error in fifo read\n");
        return;
    }

    bytes_read = uint16_val(rx, 0);
    n_samples = bytes_read / MPU6000_SAMPLE_SIZE;

    if (n_samples == 0) {
        /* Not enough data in FIFO */
        return;
    }
    [...]
```

MPU6050 (3/3)

```
    [...]
    if (n_samples > MPU6000_MAX_FIFO_SAMPLES) {
        /* Too many samples, do a FIFO RESET */
        _fifo_reset();
        return;
    }

    if (!_block_read(MPUREG_FIFO_R_W, rx, n_samples * MPU6000_SAMPLE_SIZE)) {
        hal.console->printf("MPU60x0: error in fifo read %u bytes\n",
                            n_samples * MPU6000_SAMPLE_SIZE);
        return;
    }

    _accumulate(rx, n_samples);
}
```

IMU Heating system

- Simple resistor connected to a pwm output
- Variation of the duty cycle to adjust heating power
- PID control with the temperature captured by the IMU

```
void HeatPwm::set_imu_temp(float current)
{
    float error, output;
    if (AP_HAL::millis() - _last_temp_update < 5) {
        return;
    }
    /* minimal PI algo without dt */
    error = _target - current;
    /* Don't accumulate errors if the integrated error is superior
     * to the max duty cycle(pwm_period)
     */
    if ((fabsf(_sum_error) * _Ki < _period_ns)) {
        _sum_error = _sum_error + error;
    }
    output = _Kp * error + _Ki * _sum_error;
    if (output > _period_ns) {
        output = _period_ns;
    } else if (output < 0) {
        output = 0;
    }
    _pwm->set_duty_cycle(output);
    _last_temp_update = AP_HAL::millis();
}
```

Other sensors

Other sensors

Compass

- Measures the magnetic field of the earth in its coordinates
- Determination of the orientation
- Needs calibration to determine the offsets in each direction
- AK8963 driver already implemented as a slave on MPU9250
- Adaptation of the driver for direct connection

Barometer

- Gives raw pressure (and temperature)
- Register descriptions are the same on both
- Different resolutions
  - Add support for a different resolution
  - Make the MS5611 class generic
  - Implement 2 variants for the calculation of the resolution

Motor Controller

Motor Controller

Motor Controller(1/2)

- Microcontroller that runs the motors control loop
- Connected on i2c-1
- Has its own protocol
- `https://wiki.paparazziuav.org/wiki/Bebop/BLDC`
- Original RCOutput class gives pwm values
- Transformation of PWM to RPM values

Motor Controller(2/2)

```
void RCOutput_Bebop::_set_ref_speed(uint16_t rpm[BEBOP_BLDC_MOTORS_NUM])
{
    struct bldc_ref_speed_data data;
    int i;

    data.cmd = BEBOP_BLDC_SETREFSPEED;

    for (i=0; i<BEBOP_BLDC_MOTORS_NUM; i++)
        data.rpm[i] = htobe16(rpm[i]);

    data.enable_security = 0;
    data.checksum = _checksum((uint8_t *) &data, sizeof(data) - 1);

    if (!_i2c_sem->take(0))
        return;

    hal.i2c1->write(BEBOP_BLDC_I2C_ADDR, sizeof(data), (uint8_t *)&data);

    _i2c_sem->give();
}
```

Remote Controller

Remote Controller

Remote Controller

- Ardupilot meant to be used with an RC controller
- This controller gives PWM values
- The Bebop only has a Wi-Fi connection
- Very simple protocol implemented to send PWM values

```
struct __attribute__((packed)) rc_udp_packet {
  uint32_t version;
  uint64_t timestamp_us;
  uint16_t sequence;
  uint16_t pwms[RCINPUT_UDP_NUM_CHANNELS];
};
```

- Simple utility developed to implement the remote side
- `https://github.com/jberaud/joystick_remote`

joystick_remote(1/3)

```
while (1) {
        /* wait for an event on the joystick, no timeout */
        ret = poll(&pollfd, 1, -1);
        if (ret == -1) {
            perror("joystick_thread - poll");
            break;
        } else if (ret == 0) {
            fprintf(stderr, "joystick_thread : unexpected timeout\n");
            break;
        } else if (pollfd.revents & POLLHUP) {
            fprintf(stderr, "joystick disconnected\n");
            break;
        }
        ret = read(joystick->fd, &event, sizeof(event));
        if (ret < 0) {
            perror("joystick_thread - read\n");
            break;
        }

[...]
```

joystick_remote(2/3)

```
[...]

        /* remove init flag in order not to differentiate between
         * initial virtual events and joystick events */
        event.type &= ~JS_EVENT_INIT;

        switch (event.type) {
        case JS_EVENT_AXIS:
            joystick_handle_axis(joystick, event.number, event.value);
            break;
        case JS_EVENT_BUTTON:
            joystick_handle_button(joystick, event.number, event.value);
            break;
        default:
            fprintf(stderr, "joystick_thread : unexpected event %d\n", event.type);
        }
    }
```

joystick_remote(2/3)

```
void remote_send_pwms(struct remote *remote, uint16_t *pwms,
                      uint8_t len, uint64_t micro64)
{
    static struct rc_udp_packet msg;
    int ret;

    /* to check compatibility */
    msg.version = RCINPUT_UDP_VERSION;
    msg.timestamp_us = micro64;
    msg.sequence++;

    if (len > sizeof(msg.pwms)) {
        fprintf(stderr, "remote_send_pwms : bad len %d\n", len);
        return;
    }
    memcpy(&msg.pwms, pwms, len);
    ret = sendto(remote->fd, &msg, sizeof(msg), 0,
            remote->res->ai_addr, remote->res->ai_addrlen);
    if (ret == -1) {
        perror("remote_send_pwms - socket");
        return;
    }
    return;
}
```

Ground Control Station

Ground Control Station

Ground Control Station

- Mission Planner
  - http://ardupilot.org/planner/docs/mission-planner-overview.html
- APM Planner
  - http://ardupilot.org/planner/docs/mission-planner-overview.html

- Qgroundcontrol
  - http://qgroundcontrol.org/

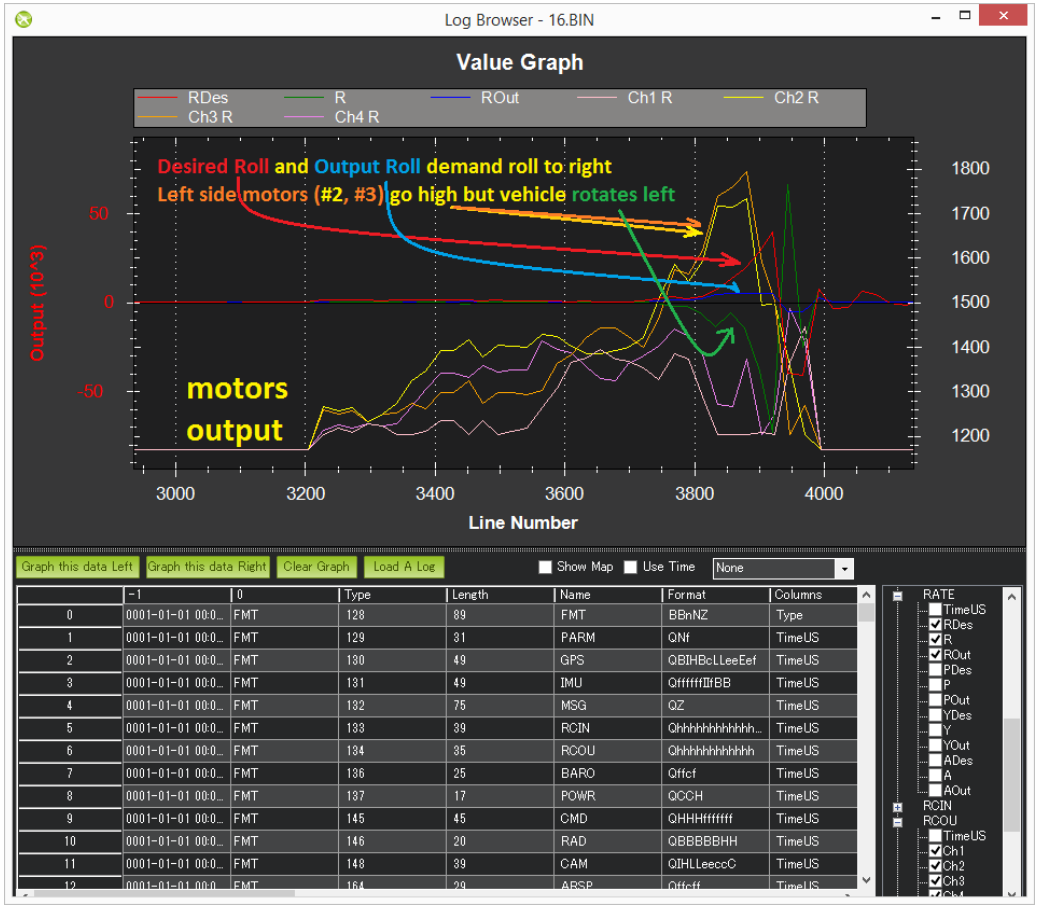
- MAVProxy
  - http://dronecode.github.io/MAVProxy/html/index.html

Parrot

First flights

First flight

- First flight = first crash
- Logging system
- Log Analysis



	-1	0	Type	Length	Name	Format	Columns
0	0001-01-01 00:0...	FMT	128	89	FMT	BBnNZ	Type
1	0001-01-01 00:0...	FMT	129	31	PARM	QNf	TimeUS
2	0001-01-01 00:0...	FMT	130	49	GPS	QBIHBcLLeeEef	TimeUS
3	0001-01-01 00:0...	FMT	131	49	IMU	QffffffIfBB	TimeUS
4	0001-01-01 00:0...	FMT	132	75	MSG	QZ	TimeUS
5	0001-01-01 00:0...	FMT	133	39	RCIN	Qhhhhhhhhhhhh...	TimeUS
6	0001-01-01 00:0...	FMT	134	35	RCOU	Qhhhhhhhhhhh	TimeUS
7	0001-01-01 00:0...	FMT	136	25	BARO	Qffcf	TimeUS
8	0001-01-01 00:0...	FMT	137	17	POWR	QCCH	TimeUS
9	0001-01-01 00:0...	FMT	145	45	CMD	QHHHfffffff	TimeUS
10	0001-01-01 00:0...	FMT	146	20	RAD	QBBBBBHH	TimeUS
11	0001-01-01 00:0...	FMT	148	39	CAM	QIHLLeeccC	TimeUS
12	0001-01-01 00:0...	FMT	164	29	ABSP	Qffcff	TimeUS

Parrot

Second flight

- `https://www.youtube.com/watch?v=hqVhh7ZxM4A`
- This crash hasn't been caused by a software bug
- It turns out I had to learn how to pilot without an automatic position control
- `http://ardupilot.org/copter/docs/flight-modes.html`

Running Ardupilot on a Bebop 2

Running Ardupilot on a Bebop 2

Toolchain and source code

Toolchain and source code

Toolchain and Source Code

Toolchain for ubuntu or debian (from jessie)

sudo apt-get install gcc-arm-linux-gnueabihf g++-arm-linux-gnueabihf

Toolchain for other distros

https://releases.linaro.org/14.07/components/toolchain/binaries/gcc-linaro-arm-linux-gnueabihf-4.9-2014.07_linux.tar.bz2

Source Code

git clone https://github.com/ArduPilot/ardupilot.git

Building

cd ardupilot/ArduCopter
make bebop
arm-linux-gnueabihf-strip ArduCopter.elf -o arducopter

Connecting and uploading the firmware

Connecting and uploading the firmare

Install adb (Android Debug Bridge)

```
sudo apt-get install android-tools-adb
```

Turn on your Bebop

Connect to its wifi network called BebopDrone2-XXXX

Enable adb by pressing the power button 4 times

Connect via adb

```
adb connect 192.168.42.1:9050
```

Push the arducopter binary to /usr/bin

```
adb shell mount -o remount,rw /
adb push arducopter /usr/bin/
```

Running ardupilot

Killing the regular autopilot and running Ardupilot

Kill the regular autopilot

```
adb shell
kk
```

Run ardupilot

```
arducopter -A udp:192.168.42.255:14550:bcast -B /dev/ttyPA1 -C
udp:192.168.42.255:14551:bcast -l /data/ftp/internal_000/APM/logs -t
/data/ftp/internal_000/APM/terrain
```

Configuring the init system to run Ardupilot at startup

Make a copy of the startup script

```
cp /etc/init.d/rcS_mode_default /etc/init.d/rcS_mode_default_backup
```

Replace the startup command

```
vi /etc/init.d/rcS_mode_default

#DragonStarter.sh -out2null &

arducopter -A udp:192.168.42.255:14550:bcast -B /dev/ttyPA1 -C
udp:192.168.42.255:14551:bcast -l /data/ftp/internal_000/APM/logs -t
/data/ftp/internal_000/APM/terrain &
```

Sync and reboot

```
sync
reboot
```

MAVProxy

MAVProxy

Install MAVProxy

```
sudo apt-get install python-matplotlib python-serial python-wxgtk2.8 python-lxml
sudo apt-get install python-scipy python-opencv python-pip python-pexpect
sudo apt-get install pymavlink MAVProxy
```

Connect to your Bebop's Wi-Fi network

Launch MAVProxy

```
mavproxy.py –master 0.0.0.0:14550 –aircraft Bebop2 –load console
```

Live Telemetry

- Init file for shortcuts `https://github.com/Dronecode/MAVProxy/blob/master/windows/mavinit.scr`
- status visible on console
- params can be seen
- "graph" command to plot telemetry logs
- Calibration of the accelerometer : "accelcal"
- Calibration of the compass : "magcal"
- Graph RC Input to see if the connection is visible

Piloting

Remote over UDP

Clone and build joystick_remote

```
git clone https://github.com/jberaud/joystick_remote
cd joystick_remote
make
```

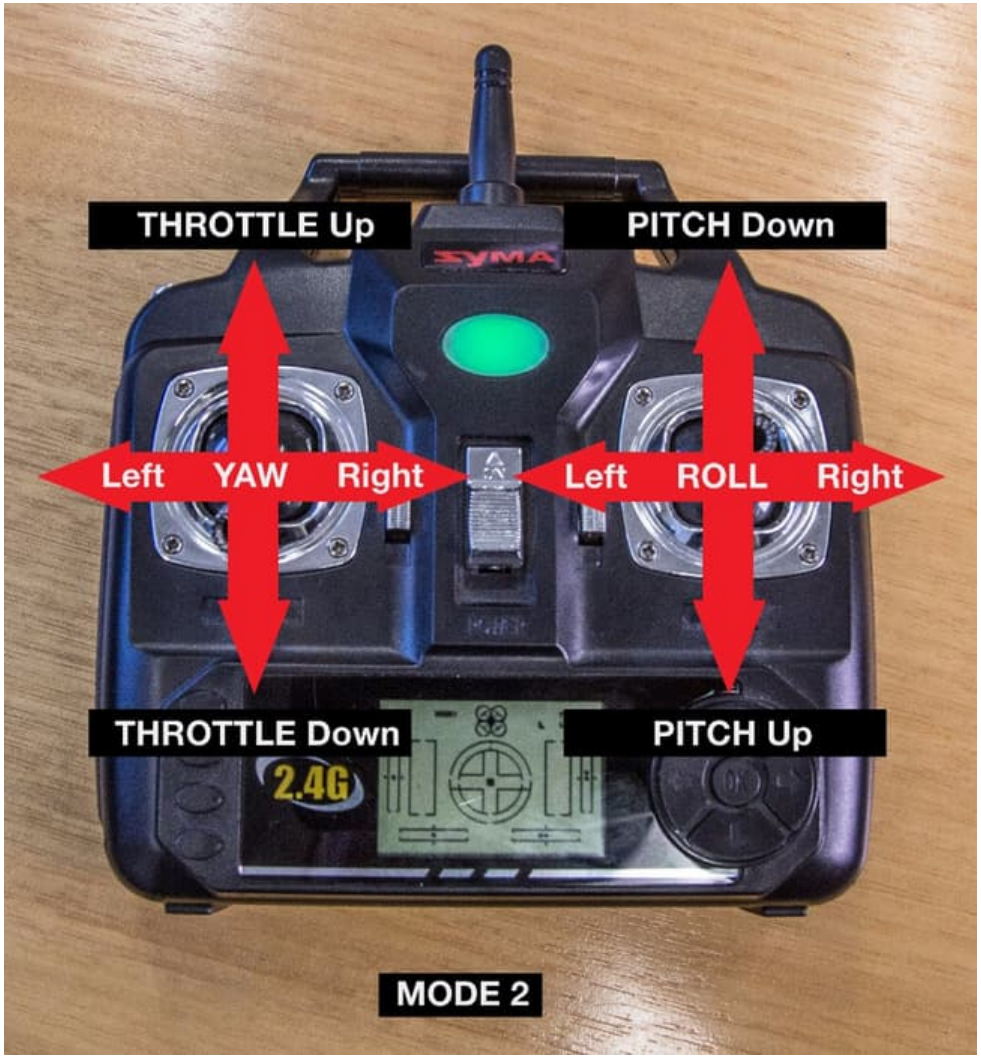
Plug your joystick (xbox360 for instance)

Launch it and check status

```
./joystick_remote -t xbox_360 -r 192.168.42.1:777
```

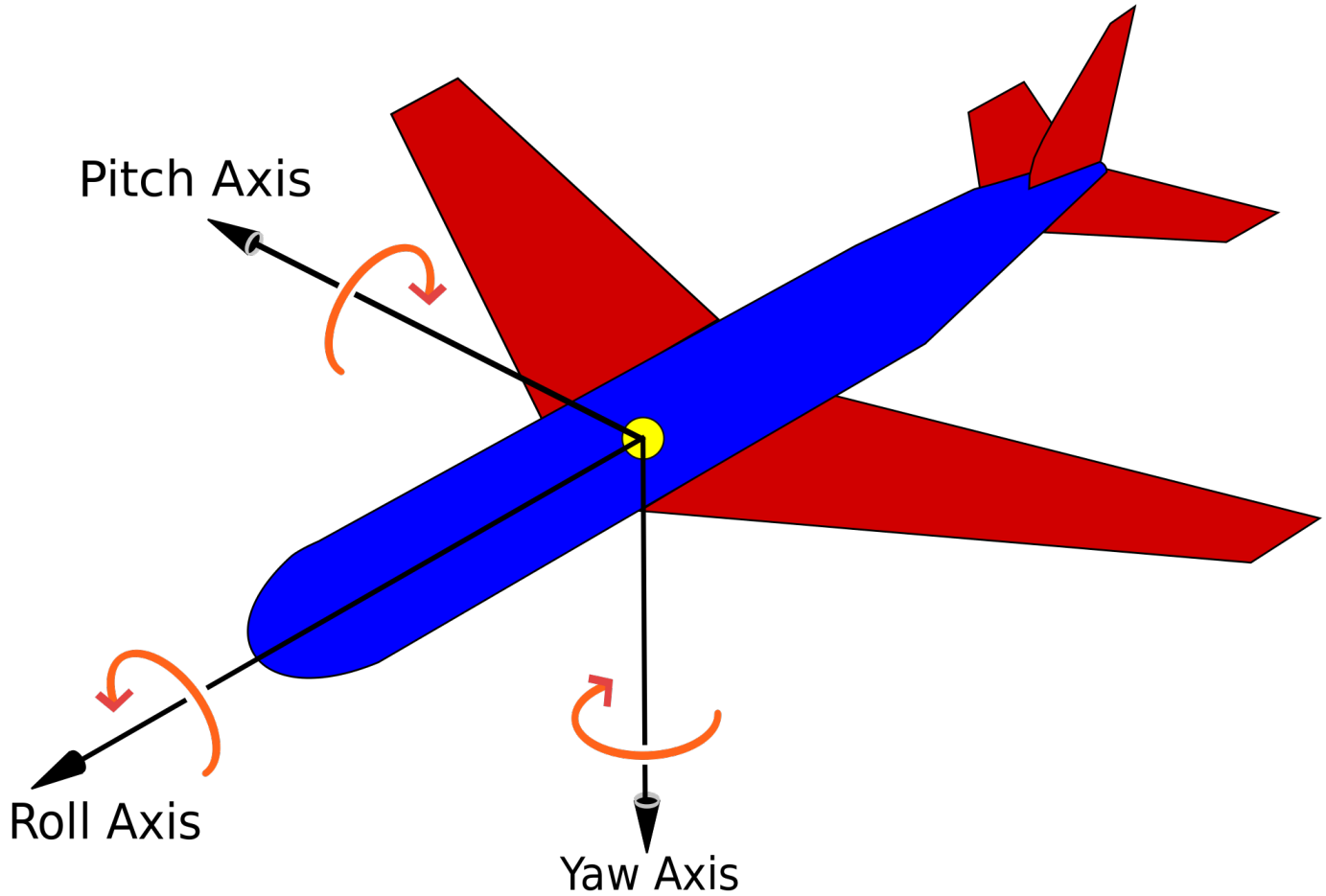
Set params for flight modes FLTMODE1 FLTMODE2 ...

Basic Piloting



Parrot

Roll Pitch Yaw



Parrot

Log Analysis

Log Analysis

- In log directory files named N.BIN
- LASTLOG.TXT gives the number of the last log
- Contain logs according to param LOG_BITMASK
- Can be analyzed with ground control stations
- MAVExplorer N.BIN
- graph command

Optical Flow

Optical Flow

- Existing solutions for optical flow on ardupilot
- Using the external PX4 optical flow module
- `https://pixhawk.org/modules/px4flow`
- PX4Flow has its own IMU and Sonar
- Doing the video frame analysis internally
- Interface to get rates over i2c

```
typedef struct i2c_integral_frame
{
    uint16_t frame_count_since_last_readout;
    int16_t pixel_flow_x_integral;
    int16_t pixel_flow_y_integral;
    int16_t gyro_x_rate_integral;
    int16_t gyro_y_rate_integral;
    int16_t gyro_z_rate_integral;
    uint32_t integration_timespan;
    uint32_t sonar_timestamp;
    int16_t ground_distance;
    int16_t gyro_temperature;
    uint8_t quality;
} __attribute__((packed)) i2c_integral_frame;
```

Optical Flow on Linux

- v4l2 capture interface
- Use already available gyro datas
  - Already unbiased by EKF
- Make it available to any ardupilot enabled Linux board
  - Generic code can be used with any usb camera

Optical Flow Inputs and Outputs

- `https://pixhawk.org/_media/modules/px4flow_paper.pdf`
- `https://github.com/PX4/Flow`
- Inputs
  - 2 images
  - Corresponding angular speeds
  - Sensor/Lens dimensions and parameters
- Outputs
  - Delta angular speed
  - Delta time
  - Delta angular speed from gyros over the same delta time

Implementation on Linux

mt9v117 sensor configuration over i2c

- PWM for the sensor's master clock
- GPIO userland driver for reset PIN
- v4l2-subdev driver available but not included in official kernel
- dummy v4l2-subdev driver for compatibility
- Userspace driver over i2c-dev
- Static configuration done at startup
- Setting it to run at the maximum framerate : 89.2fps on the Bebop

Capture using v4l2

```
class Linux::VideoIn {
public:
    /* This structure implements the fields of the v4l2_pix_format struct
     * that are considered useful for an optical flow application along
     * with the v4l2_buffer fields timestamp and sequence*/
    class Frame {
    friend class VideoIn;
    public:
        uint32_t timestamp;
        uint32_t sequence;
        void *data;
    private:
        uint32_t buf_index;
    };

    bool get_frame(Frame &frame);
    void put_frame(Frame &frame);
    void set_device_path(const char* path);
    void init();
    bool open_device(const char *device_path, uint32_t memtype);
    bool allocate_buffers(uint32_t nbufs);
    bool set_format(uint32_t *width, uint32_t *height, uint32_t *format,
                    uint32_t *bytesperline, uint32_t *sizeimage);
    bool set_crop(uint32_t left, uint32_t top,
                  uint32_t width, uint32_t height);
    void prepare_capture();

private:
    [...]
};
```

Flow algorithm

Flow algorithm

Distance between 2 images in pixels(1/2)

- Sum of Average Differences

```
static inline uint32_t compute_sad(uint8_t *image1, uint8_t *image2,
                                   uint16_t off1x, uint16_t off1y,
                                   uint16_t off2x, uint16_t off2y,
                                   uint16_t row_size, uint16_t window_size)
{
    /* calculate position in image buffer
     * off1 for image1 and off2 for image2
     */
    uint16_t off1 = off1y * row_size + off1x;
    uint16_t off2 = off2y * row_size + off2x;
    unsigned int i,j;
    uint32_t acc = 0;

    for (i = 0; i < window_size; i++) {
        for (j = 0; j < window_size; j++) {
            acc += abs(image1[off1 + i + j*row_size] -
                       image2[off2 + i + j*row_size]);
        }
    }
    return acc;
}
```

Distance between 2 images in pixels(1/2)

- 8x8 blocks
- For each block in image 1 (x1, y1)
  - Calculate SAD with blocks in image 2 (x2, y2)
  - from x2 = x1 - 4 to x2 = x1 + 4
  - from y2 = y1 - 4 to y2 = y1 + 4
- See which translation minimizes the SAD
- For N blocks in image 1
- Calculate the average translation

```
for (jj = winmin; jj <= winmax; jj++) {
    for (ii = winmin; ii <= winmax; ii++) {
        uint32_t temp_dist = compute_sad(image1, image2, i, j,
                                         i + ii, j + jj,
                                         (uint16_t)_bytesperline,
                                         2 * _search_size);

        if (temp_dist < dist) {
            sumx = ii;
            sumy = jj;
            dist = temp_dist;
        }
    }
}
```

From distance in pixels to angular speed

- focal length of the camera module : 2.5mm
- pixel size : 3.6µm
- binning : x2 in each direction
- crop/rescale : 240 pixels resized in 64

flow calculation

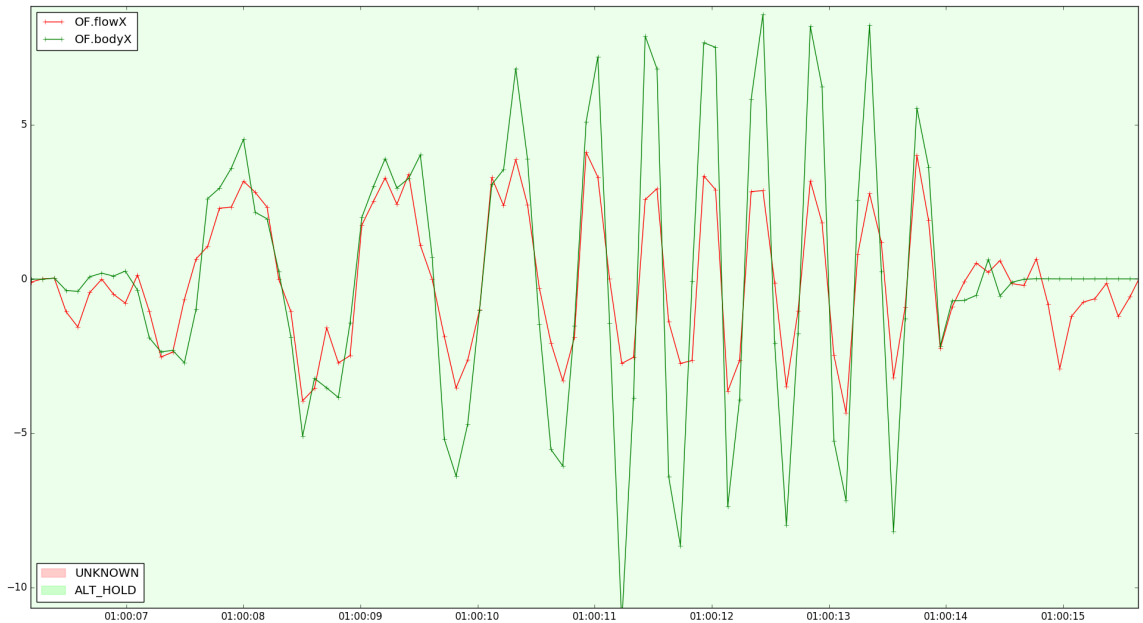
$$flow\_x\_radians = \frac{flow\_x\_pixels}{focal\_length\_pixels}$$

$$focal\_length\_pixels = \frac{2500}{3.6 \times 2 \times 240/64}$$

Ardupilot integration

Ardupilot Integration

- Optical Flow interface added in HAL
- Other optical flow backend
- Tests with dataflash logs
- Roll/Pitch without translation
- Compare with gyro data to validate angular rates

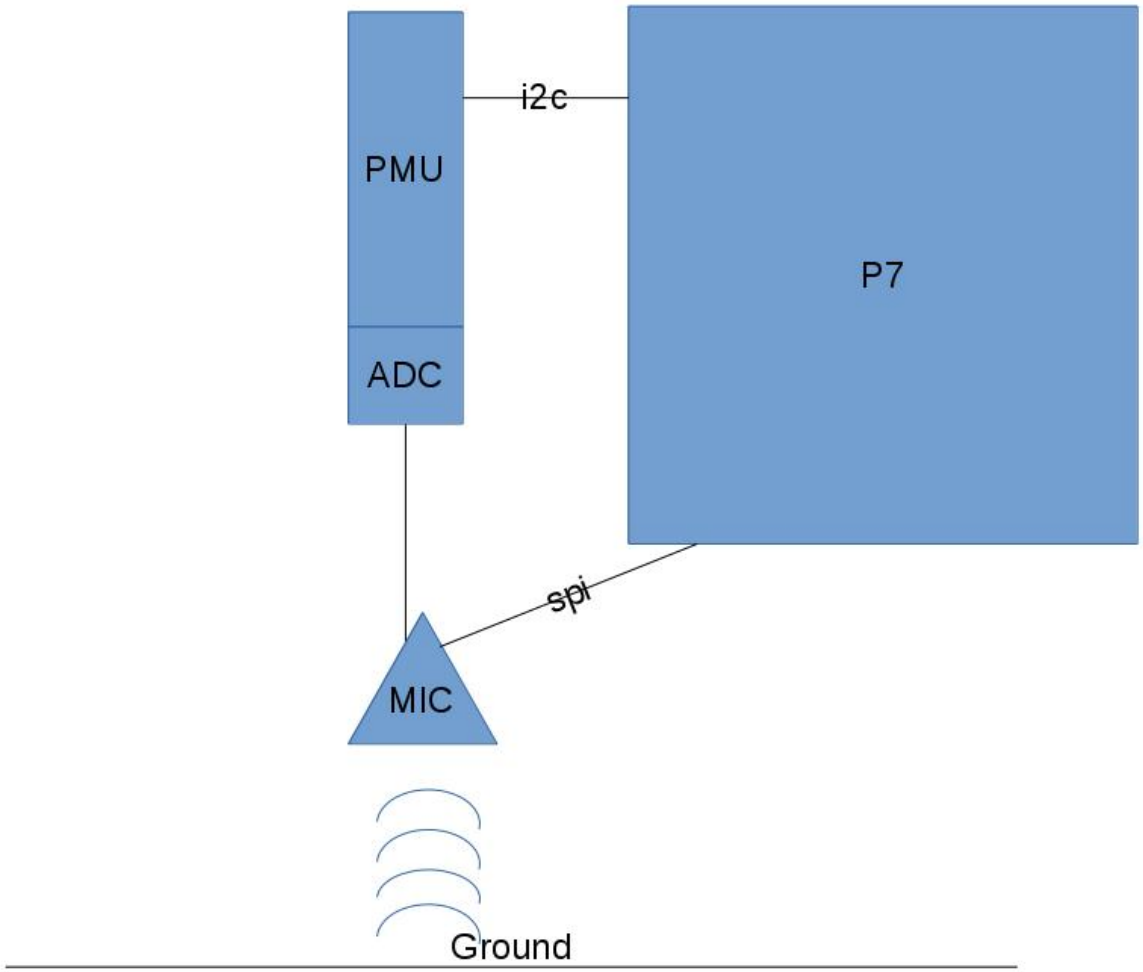


Parrot

Sonar

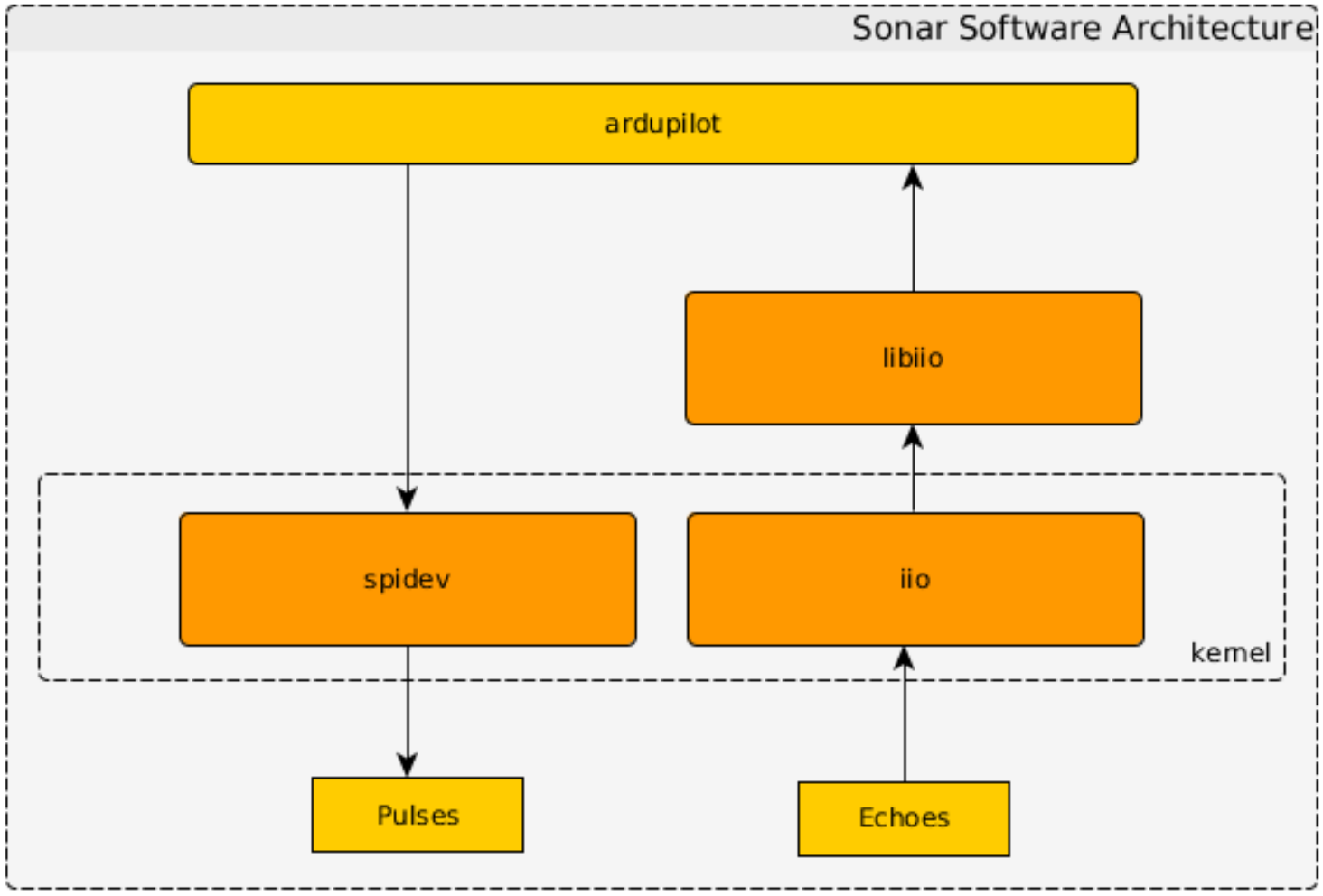
Sonar

Hardware architecture



Parrot

Sonar software architecture



Parrot

Sonar pulses capture

```
int UltraSound_Bebop::configure_capture()
{
    const char *adcname = "p7mu-adc_2";
    char *adcchannel = "voltage2";
    /* configure adc interface using libiio */
    _iio = iio_create_local_context();
    if (!_iio)
        return -1;
    _adc.device = iio_context_find_device(_iio, adcname);
    if (!_adc.device) {
        goto error_destroy_context;
    }
    _adc.channel = iio_device_find_channel(_adc.device, adcchannel,
            false);
    if (!_adc.channel) {
        goto error_destroy_context;
    }

    iio_channel_enable(_adc.channel);
    [...]
```

Sonar pulses capture(2/2)

```
    [...]
    _adc.freq = P7_US_DEFAULT_ADC_FREQ >> P7_US_FILTER_POWER;
    _adc.threshold_time_rejection = 2.0 / P7_US_SOUND_SPEED *
        _adc.freq;

    /* Create input buffer */
    _adc.buffer_size = P7_US_P7_COUNT;
    if (iio_device_set_kernel_buffers_count(_adc.device, 1)) {
        goto error_destroy_context;
    }
    _adc.buffer = iio_device_create_buffer(_adc.device,
            _adc.buffer_size, false);
    if (!_adc.buffer) {
        goto error_destroy_context;
    }

    return 0;

error_buffer_destroy:
    iio_buffer_destroy(_adc.buffer);
    _adc.buffer = NULL;
error_destroy_context:
    iio_context_destroy(_iio);
    _iio = NULL;
    return -1;
}
```

Sonar pulses

```
int UltraSound_Bebop::launch()
{
    iio_device_attr_write(_adc.device, "buffer/enable", "1");
    _spi->transfer(_tx_buf, P7_US_NB_PULSES_MAX);
    return 0;
}

int UltraSound_Bebop::capture()
{
    int ret;

    ret = iio_buffer_refill(_adc.buffer);
    iio_device_attr_write(_adc.device, "buffer/enable", "0");
    return ret;
}
```

Altitude calculation

```
while(1) {
    _ultrasound->launch();

    _ultrasound->capture();
    _adcCapture = _ultrasound->get_capture();

    if (applyAveragingFilter() < 0) {
        LOGW("Could not apply averaging filter");
        goto endloop;
    }

    if (searchLocalMaxima() < 0) {
        LOGW("Did not find any local maximum");
        goto endloop;
    }

    maxIndex = searchMaximumWithMaxAmplitude();
    if (maxIndex >= 0) {
        _altitude = (float)(maxIndex * P7_US_SOUND_SPEED) /
                (2 * (P7_US_DEFAULT_ADC_FREQ / _filterAverage));
        _mode = _ultrasound->update_mode(_altitude);
    }
}
```

Monitoring real-time performances with LTTng

Real time issues

- Real time issues encountered when porting ardupilot
- Enabling param SCHED_DEBUG shows statistics about execution time
- The main loop is supposed to last 2.5ms
- Every 10s SCHED_DEBUG outputs the number of loops above this limit
- It also displays the maximum and minimum time spent in a loop
- PERF: 3/1000 3100 1402

LTTng

- LTTng is a tracing tool
- Tracing : Recording the real time behaviour of a software
- Analyze the recorded datas off-line
- LTTng can be used to analyze both the kernel and userland applications
- liblttng-ust : library for userland tracing
- `http://lttng.org/docs/`

tracepoint events declaration

```
TRACEPOINT_EVENT(
    ardupilot,
    begin,
    TP_ARGS(
        char*, name_arg
    ),
    TP_FIELDS(
        ctf_string(name_field, name_arg)
    )
)

TRACEPOINT_EVENT(
    ardupilot,
    end,
    TP_ARGS(
        char*, name_arg
    ),
    TP_FIELDS(
        ctf_string(name_field, name_arg)
    )
)
```

tracepoint events usage

```
void Perf_Lttng::begin()
{
    if (_type != AP_HAL::Util::PC_ELAPSED) {
        return;
    }
    tracepoint(ardupilot, begin, _name);
}

void Perf_Lttng::end()
{
    if (_type != AP_HAL::Util::PC_ELAPSED) {
        return;
    }
    tracepoint(ardupilot, end, _name);
}
```

Using Perf Class in ardupilot

```
/* create perf object */
_perf_FuseOptFlow(hal.util->perf_alloc(AP_HAL::Util::PC_ELAPSED, "EK2_FuseOptFlow"));

/* begin perf */
hal.util->perf_begin(_perf_FuseOptFlow);

/* end perf */
hal.util->perf_end(_perf_FuseOptFlow);
```

Enabling LTTng events at runtime

Enumerate available events

lttng list –userspace
ardupilot:count (loglevel: TRACE_DEBUG_LINE (13)) (type: tracepoint)
ardupilot:end (loglevel: TRACE_DEBUG_LINE (13)) (type: tracepoint)
ardupilot:begin (loglevel: TRACE_DEBUG_LINE (13)) (type: tracepoint)

Create tracing session

lttng create -o trace_00

Enable ardupilot perf events

lttng enable-event –userspace ardupilot:end lttng enable-event –userspace ardupilot:begin

Start tracing

lttng start

Stop tracing session

lttng stop lttng destroy

Downloading and analyzing the captured trace

- lttng2lxt `https://github.com/jberaud/lttng2lxt`
- produces a file readable by gtkwave

Download the trace

```
adb pull /data/ftp/internal_000/APM/trace_00
```

Use babeltrace to translate the trace into text

```
babeltrace trace_00
```

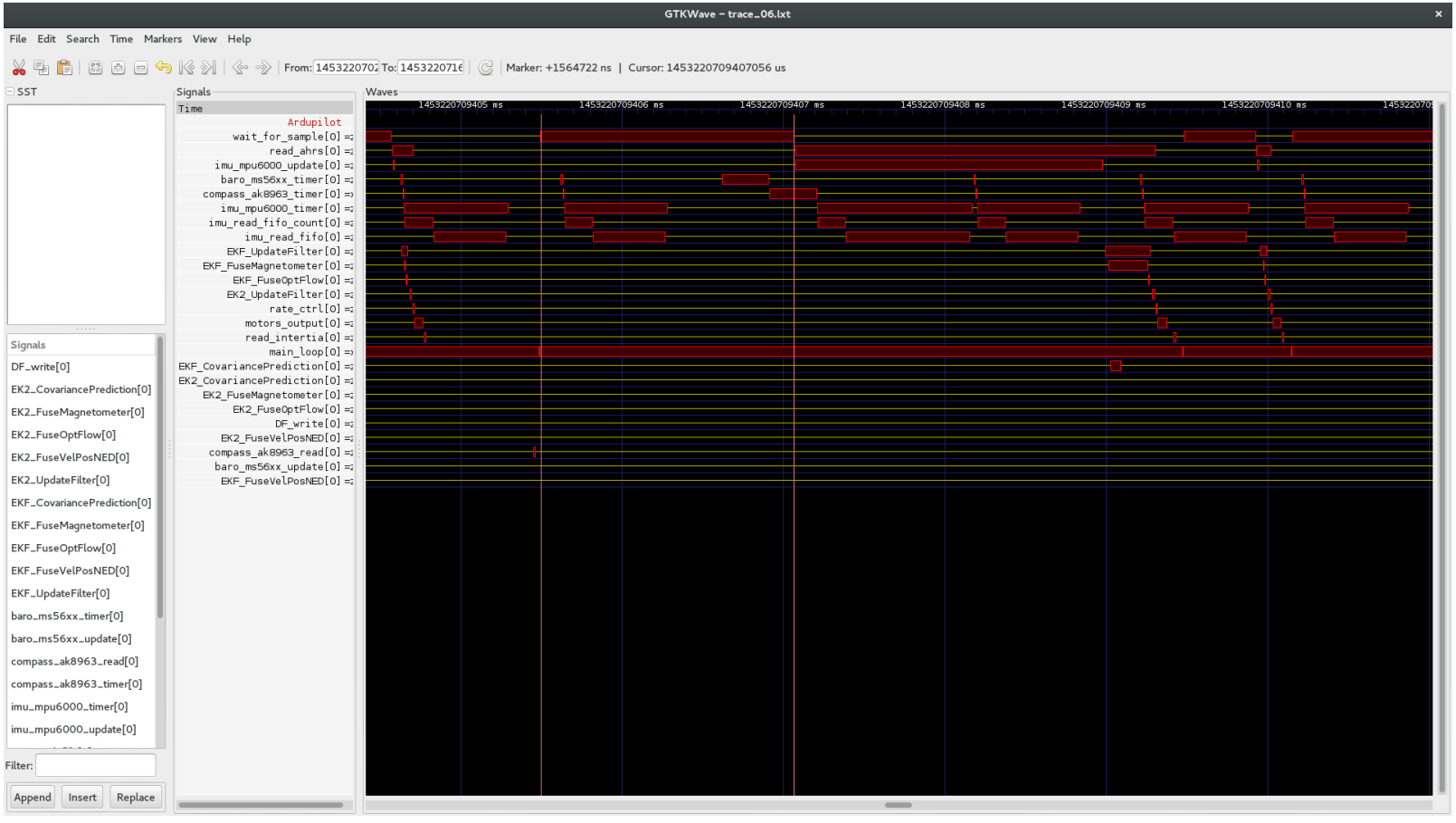
Use lttng2lxt to produce an lxt file

```
lttng2lxt trace_00
```

Using gtkwave

Launch gtkwave

gtkwave -A trace_00.lxt



Parrot

Conclusion

Conclusion

Ongoing and future work

- Finish Sonar driver and have it merged into master
- Test and improve the optical flow
- Add support for video
  - gstreamer ?
  - IPC to export datas from ardupilot to the video application
  - Fully open source solution (no digital stabilization) ?
- Integrate support for ardupilot as an alternative to our proprietary autopilot ?
- Integration in future Parrot products

References and useful links

- https://github.com/ArduPilot/ardupilot
- http://dev.ardupilot.com
- http://ardupilot.org/dev/docs/building-for-bebop-on-linux.html
- http://ardupilot.org/dev/docs/building-for-bebop-2.html
- http://ardupilot.org/dev/docs/using-linux-trace-toolkit-ng-lttng-to-trace-ardupilot-in-realtime.html

Conclusion

Questions ?