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

Julien BERAUD

April 5th 2016 - Embedded Linux Conference San Diego
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 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
Linux integration

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

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

```
root@milosboard:/ # ls /dev
LNO       media0
Magnetometer mem          tty1         tty40
Motors     mmcblk0        tty10        tty41
Pressure   mmcblk0/boot0  tty11         tty42
android_adb mmcblk0/boot1 tty12         tty43
bus        mt9f002         tty13         tty44
console    mt9v1l7         tty14         tty45
cpu_dma_latency mt0         tty15         tty46
fb         mt0ro          tty16         tty47
fd         mt1            tty17         tty48
full       mtd1           tty18         tty49
hx280a      mtd2           tty19         tty5
i2c-0       mtd2ro         tty2          tty50
i2c-1       mtd3           tty20         tty51
i2c-2       mtd3ro         tty21         tty52
i2c-akm0963 mt4            tty22         tty53
i2c-cypress mt4ro          tty23         tty54
i2c-mpu6050 mtplan         tty24         tty55
i2c-ms5607  network_latency tty25         tty56
i2c-mt9f002 network_throughput tty26         tty57
i2c-mt9v1l7 null           tty27         tty58
i2c-p7mu    p7mu           tty28         tty59
iio:device0 ptk            tty29         tty6
iio:device1 pts            tty30         tty60
iio:device2 random         tty31         tty61
iio:device3 rtc0           tty32         tty62
iio:device4 shm            tty33         tty63
iio:device5 socket         tty34         tty64
iio:device6 spidev1.0      tty35         tty65
kmem        stderr         tty36         tty66
kmsg        stdin          tty37         tty67
loop-control stdout        tty38         tty68
loop0       tty            tty39         tty69
m0          tty0           tty40         tty70

root@milosboard:/ #
```
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

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

- Accelerometer and gyroscope
- Gives a 3D acceleration vector \((x, y, z)\)
- Gives a 3D angular speed vector \((\text{roll, pitch, yaw})\)
- MPU6050 runs over i2c
- 8kHz maximum gyros and 1kHz maximum accelerometers
- 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
```c
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;
  }
  [...]
```
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

```cpp
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
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 (1/2)

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

- 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

```c
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](https://github.com/jberaud/joystick_remote)
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);

}
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

- Mission Planner
- APM Planner
- Qgroundcontrol
  - http://qgroundcontrol.org/
- MAVProxy
First flight

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

```
```

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 firmware

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Install adb (Android Debug Bridge)</strong></td>
<td><code>sudo apt-get install android-tools-adb</code></td>
</tr>
<tr>
<td><strong>Turn on your Bebop</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Connect to its wifi network called BebopDrone2-XXXX</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Enable adb by pressing the power button 4 times</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Connect via adb</strong></td>
<td><code>adb connect 192.168.42.1:9050</code></td>
</tr>
<tr>
<td><strong>Push the arducopter binary to /usr/bin</strong></td>
<td><code>adb shell mount -o remount,rw /</code> <code>adb push arducopter /usr/bin/</code></td>
</tr>
</tbody>
</table>
Running ardupilot
### Kill the regular autopilot

```
adb shell
kk
```

### Run ardupilot

```
```
Configuring the init system to run Ardupilot at startup

<table>
<thead>
<tr>
<th>Make a copy of the startup script</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cp /etc/init.d/rcS_mode_default /etc/init.d/rcS_mode_default_backup</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Replace the startup command</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vi /etc/init.d/rcS_mode_default</code></td>
</tr>
<tr>
<td><code>#DragonStarter.sh -out2null &amp;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sync and reboot</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sync</code></td>
</tr>
<tr>
<td><code>reboot</code></td>
</tr>
</tbody>
</table>
MAVProxy

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

- 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

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

THROTTLE Up

PITCH Down

Left YAW Right

Left ROLL Right

THROTTLE Down

PITCH Up

MODE 2
Roll Pitch Yaw
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
Optical Flow

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

```c
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://pixhawk.org/_media/modules/px4flow_paper.pdf)
- [https://github.com/PX4/Flow](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

```c
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 \((x_1, y_1)\)
  - Calculate SAD with blocks in image 2 \((x_2, y_2)\)
  - from \(x_2 = x_1 - 4\) to \(x_2 = x_1 + 4\)
  - from \(y_2 = y_1 - 4\) to \(y_2 = y_1 + 4\)
- See which translation minimizes the SAD
- For N blocks in image 1
- Calculate the average translation

```c
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

\[
\text{flow}_x_{\text{radians}} = \frac{\text{flow}_x_{\text{pixels}}}{\text{focal}_\text{length}_\text{pixels}}
\]

\[
\text{focal}_\text{length}_\text{pixels} = \frac{2500}{3.6 \times 2 \times 240/64}
\]
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
Hardware architecture

- PMU
- ADC
- MIC
- P7
- Ground
Sonar pulses capture

```c
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)

```c
[...] _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

```cpp
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

```c
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);
    }
}
```

```
Hacking a Commercial Drone to run an Open Source Autopilot - APM on Parrot Bebop 74 / 87
```
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

```cpp
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

```c
/* 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lttng list</code> –userspace</td>
<td>Lists available events</td>
</tr>
<tr>
<td><code>ardupilot:count (loglevel: TRACE_DEBUG_LINE (13)) (type: tracepoint)</code></td>
<td>Specific ardupilot event</td>
</tr>
<tr>
<td><code>ardupilot:end (loglevel: TRACE_DEBUG_LINE (13)) (type: tracepoint)</code></td>
<td>Specific ardupilot event</td>
</tr>
<tr>
<td><code>ardupilot:begin (loglevel: TRACE_DEBUG_LINE (13)) (type: tracepoint)</code></td>
<td>Specific ardupilot event</td>
</tr>
</tbody>
</table>

### Create tracing session

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lttng create -o trace_00</code></td>
<td>Creates a tracing session with the specified output file name</td>
</tr>
</tbody>
</table>

### Enable ardupilot perf events

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lttng enable-event –userspace ardupilot:end</code></td>
<td>Enables ardupilot event</td>
</tr>
<tr>
<td><code>lttng enable-event –userspace ardupilot:begin</code></td>
<td>Enables ardupilot event</td>
</tr>
</tbody>
</table>

### Start tracing

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lttng start</code></td>
<td>Starts tracing session</td>
</tr>
</tbody>
</table>

### Stop tracing session

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lttng stop</code></td>
<td>Stops tracing session</td>
</tr>
<tr>
<td><code>lttng destroy</code></td>
<td>Destroys the tracing session</td>
</tr>
</tbody>
</table>
Downloading and analyzing the captured trace

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

---

Download the trace

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

Use babeltrace to translate the trace into text

```bash
babeltrace trace_00
```

Use `lttng2lxt` to produce an lxt file

```bash
lttng2lxt trace_00
```
Using gtkwave

Launch gtkwave

gtkwave -A trace_00.lxt
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 data 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
Conclusion

Questions ?