



# Istrobotics

Robotour 2017, 17.9.2017

Pavol Boško, Peter Boško, Radoslav Kováč, Tomáš Kováč

2016



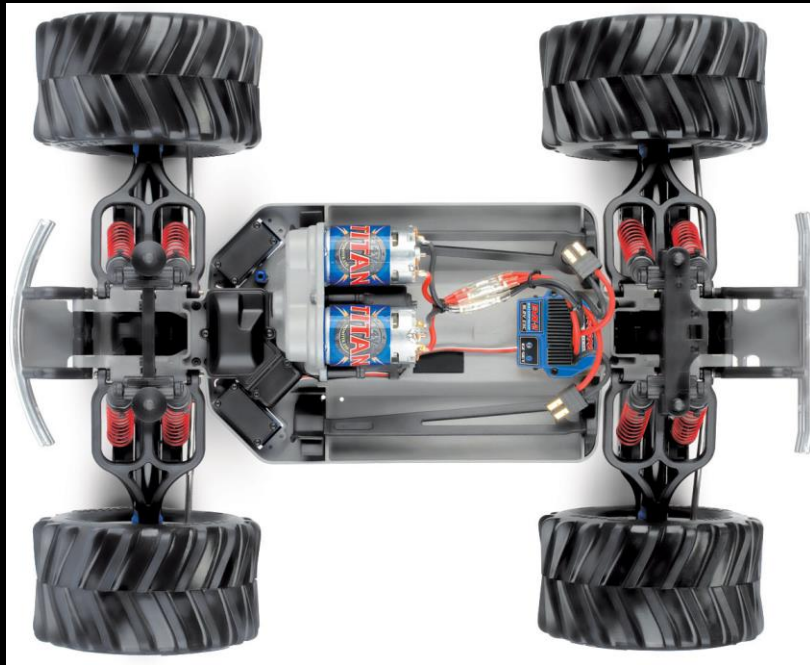
2017







# ROBOT CHASSIS

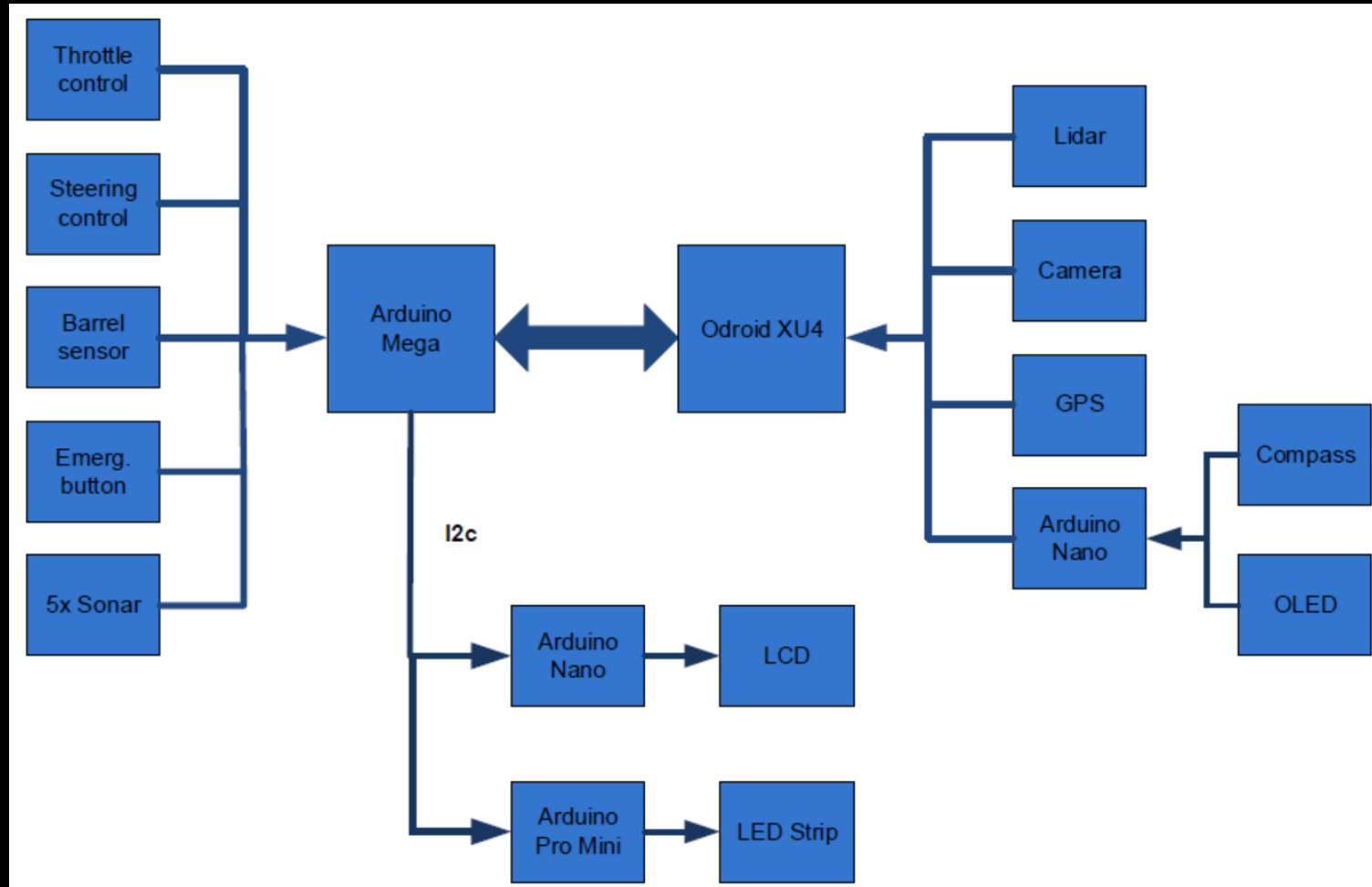


- RC model: Traxxas E-Maxx 4x4 monster truck
- Top Speed: 48 km/h
- Waterproof electronics, servos

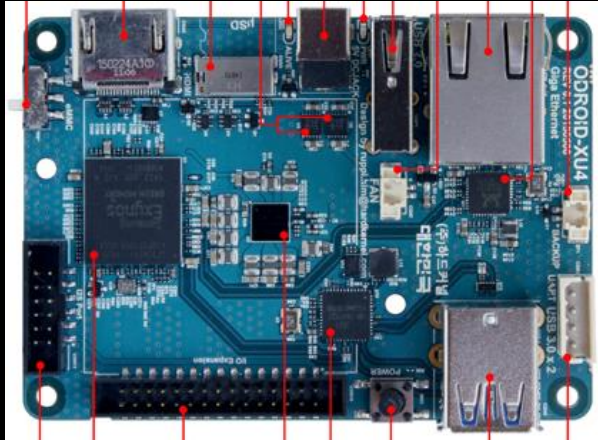




# HARDWARE DESIGN



# HARDWARE

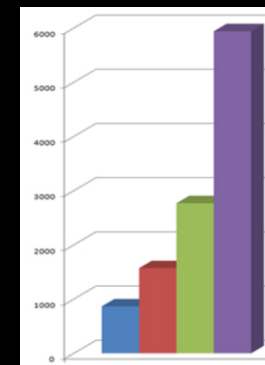


- **Odroid-XU4:** 2GHz 8-core, 2GB RAM
- Arduino Mega: 16MHz, 8KB RAM
- 2x Arduino Nano: 16MHz, 2KB RAM
- Arduino Pro Mini: 16MHz, 2KB RAM
- 2D Lidar: RoboPeak RPLIDAR 360 (\$400)
- Camera: Odroid USB Cam (640x480, 60 FOV)
- Mouse type GPS/Glonass: Holux M-215+
- Compass: Bosch BNO055
- **5x Sonar:** HC - SR04
- LCD & OLED displays, **8x LED**

# Odroid-XU4 vs Raspberry Pi3

	Raspberry Pi3	Odroid-XU4
<b>CPU</b>	ARM Cortex-A53	Samsung Exynos5422 Cortex
<b>Clock</b>	1.2 GHz	2 GHz
<b>Cores</b>	4x	8x
<b>RAM</b>	1GB LPDDR2	2GB LPDDR3
<b>Flash</b>	microSD	eMMC5.0 HS400
<b>Ethernet</b>	10/100 Mbit	1 Gigabit
<b>USB</b>	4x USB 2.0	2x USB 3.0 1x USB 2.0

	R-Pi2	R-Pi3	O-XU4
<b>Image processing</b>	164,4 ms	167,3 ms	26,5 ms
<b>JPG/PNG writing</b>	-	-	3x faster
<b>Processing lag</b>	12 sec	2 sec	100 ms



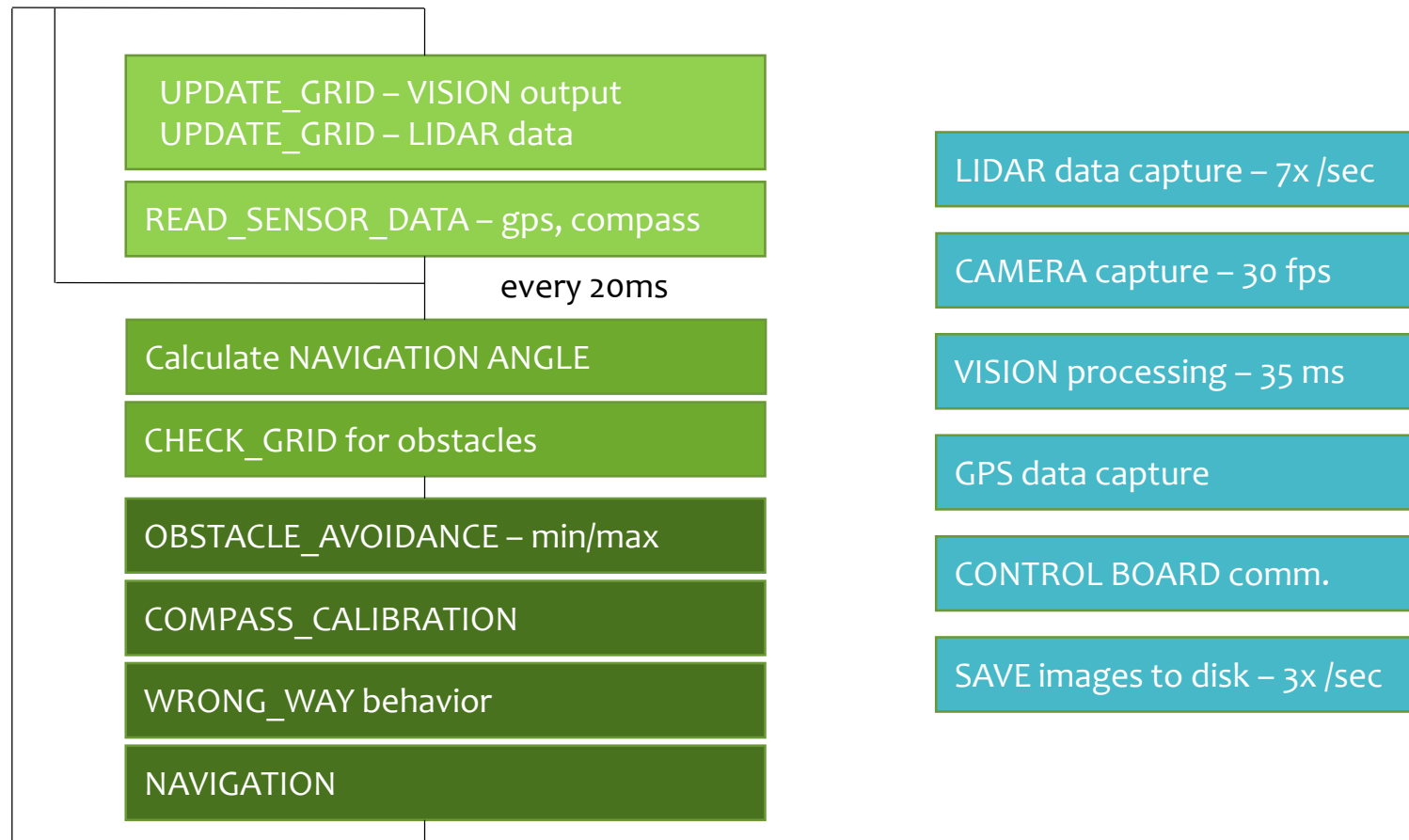
# SOFTWARE



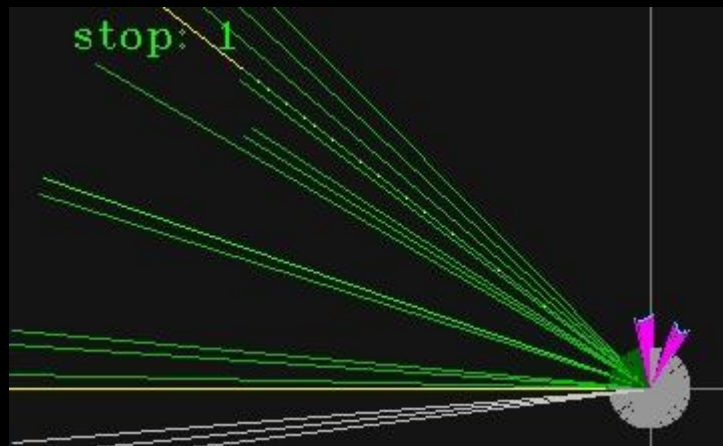
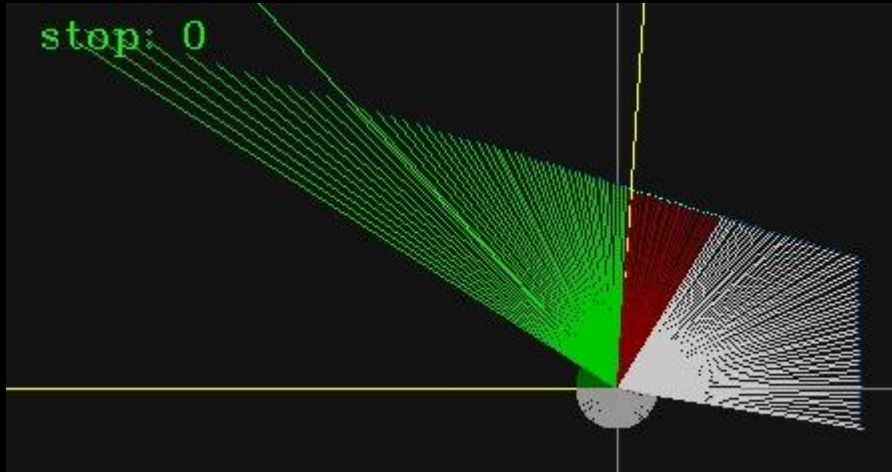
- Operating system: Ubuntu 16.04 Mate
- Source codes: C++, 340kB (2016: 180kB)
- Vision library: OpenCV
- Geo library: GeographicLib
- Logging framework: Apache log4cxx
- Main application + 8x pthreads
  - 4x sensors (Camera, Lidar, GPS , Compass)
  - image capturing + vision processing
  - output: image saving (1GB of data/ round)
  - control board (Compass)



# SOFTWARE DESIGN – PROCESSING

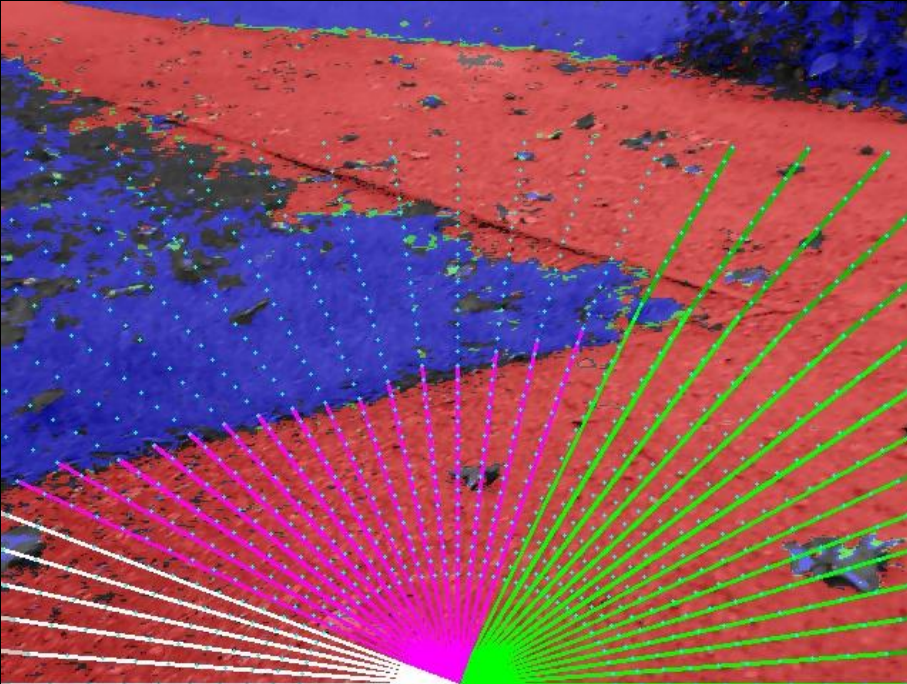


# LIDAR – obstacle detection



- Obstacle detection condition (red):
  - If distance is  $< 100$  cm
  - Filtering: distance  $< 1$ cm (grey)
- Stop condition (pink):
  - Check angle:  $-45$  to  $+45$  degrees
  - If distance is  $< 50$  cm at 3 diff. degs
  - Sonars were also used (rain issue)
- Obstacle avoidance (green/white)
  - Find OK intervals of  $> 20$  degrees
  - Choose the closest to going straight

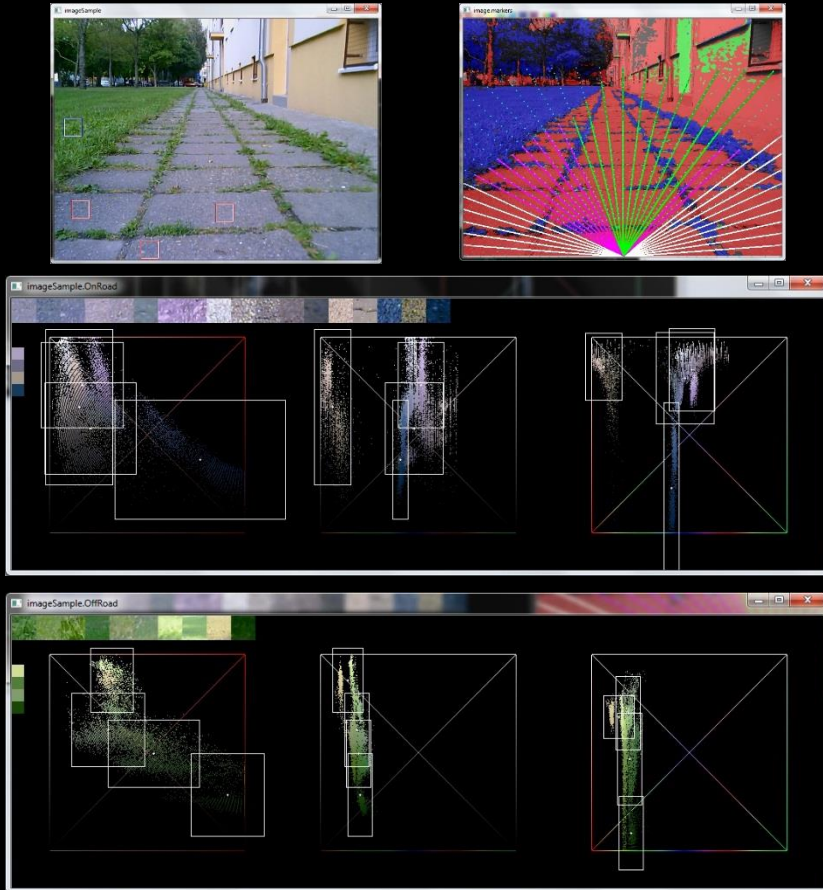
# VISION – approach



- Our approach: **lidar-like local map**
  - For any seen angle is obstacle closer than 1 meter?
  - 1 meter or to the image border
- Algorithm:
  - Pixel color classification
  - Evaluate grid points
  - Calculate distance to obstacle
  - Find OK intervals – same like LIDAR

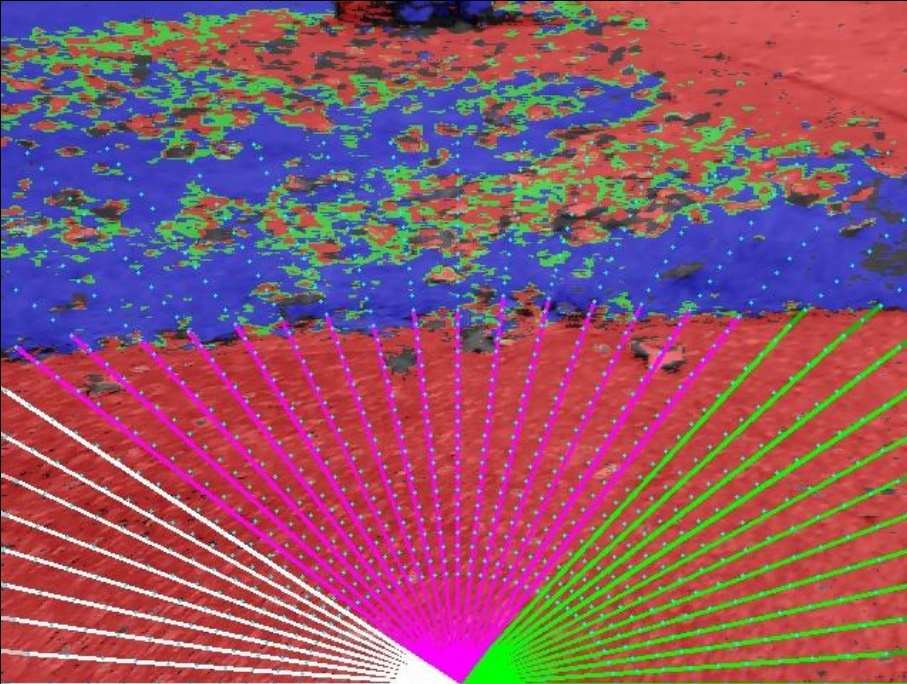


# VISION – Pixel color classification



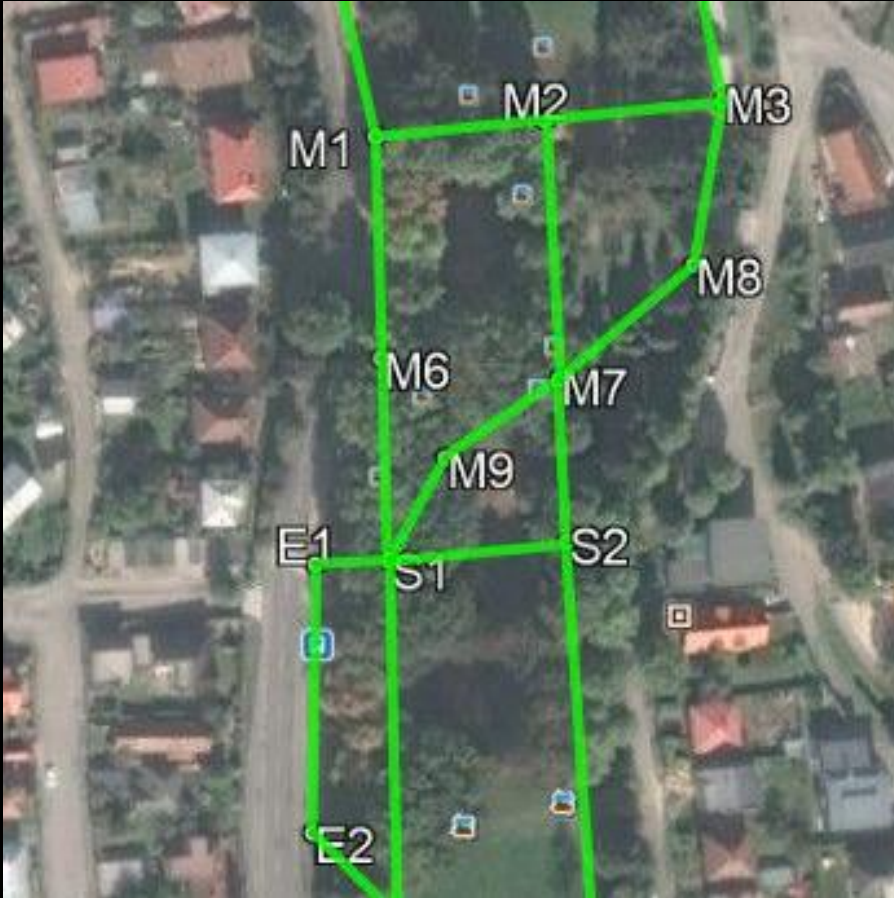
- Approach:
  - Choose sample pixel blocks (32x32) from training images
  - Calculate 4 clusters centers in color space (OpenCV kmeans)
  - Calculate cluster radius (histogram based)
  - Repeat for 2 classifiers : road and off-road (grass)
- HSV color space + Euclidian distance
- Tool was developed - to define pixel blocks and evaluate images

# VISION – Algorithm



- Pixel color classification - 4 results:
  - Road (red)
  - Off-road (blue)
  - Both (green)
  - None (grey)
- Evaluate grid points
  - Cca 1000 points in 37 lines (5 deg)
  - Evaluating nearby pixels (80x80)
  - Majority of “Road” pixels is checked
- Calculate distance to obstacle
- Find OK intervals + merge with LIDAR

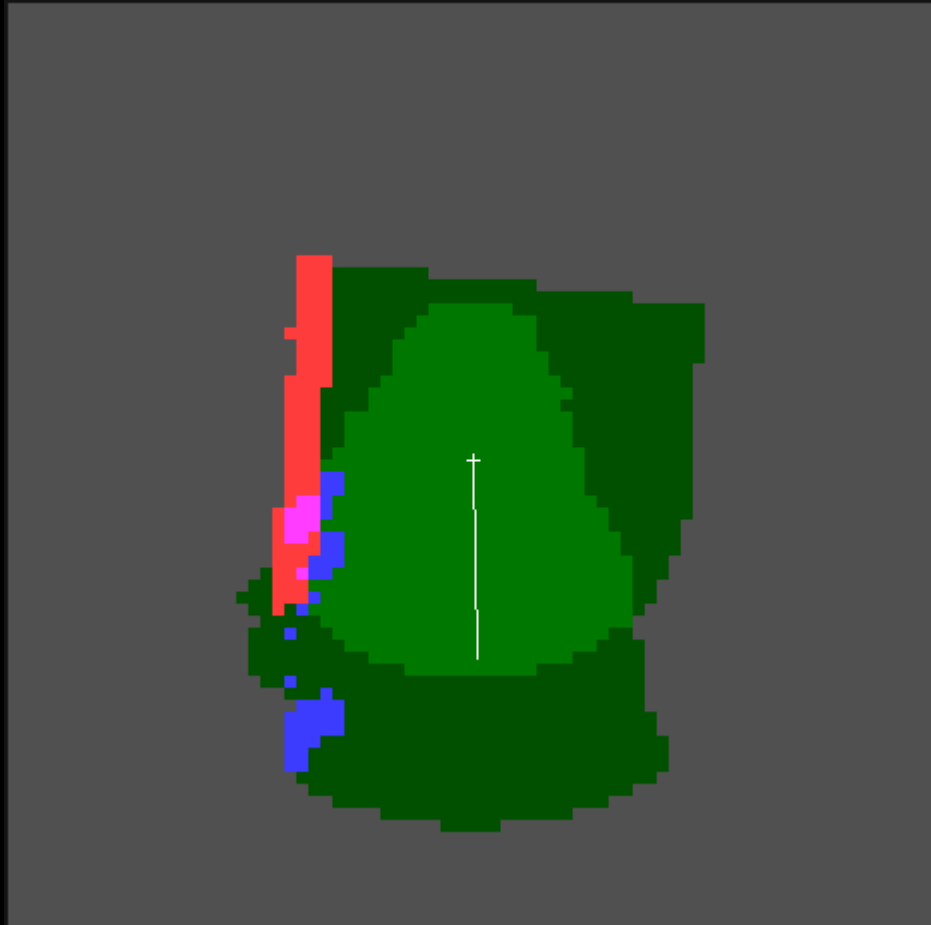
# WAYPOINT NAVIGATION



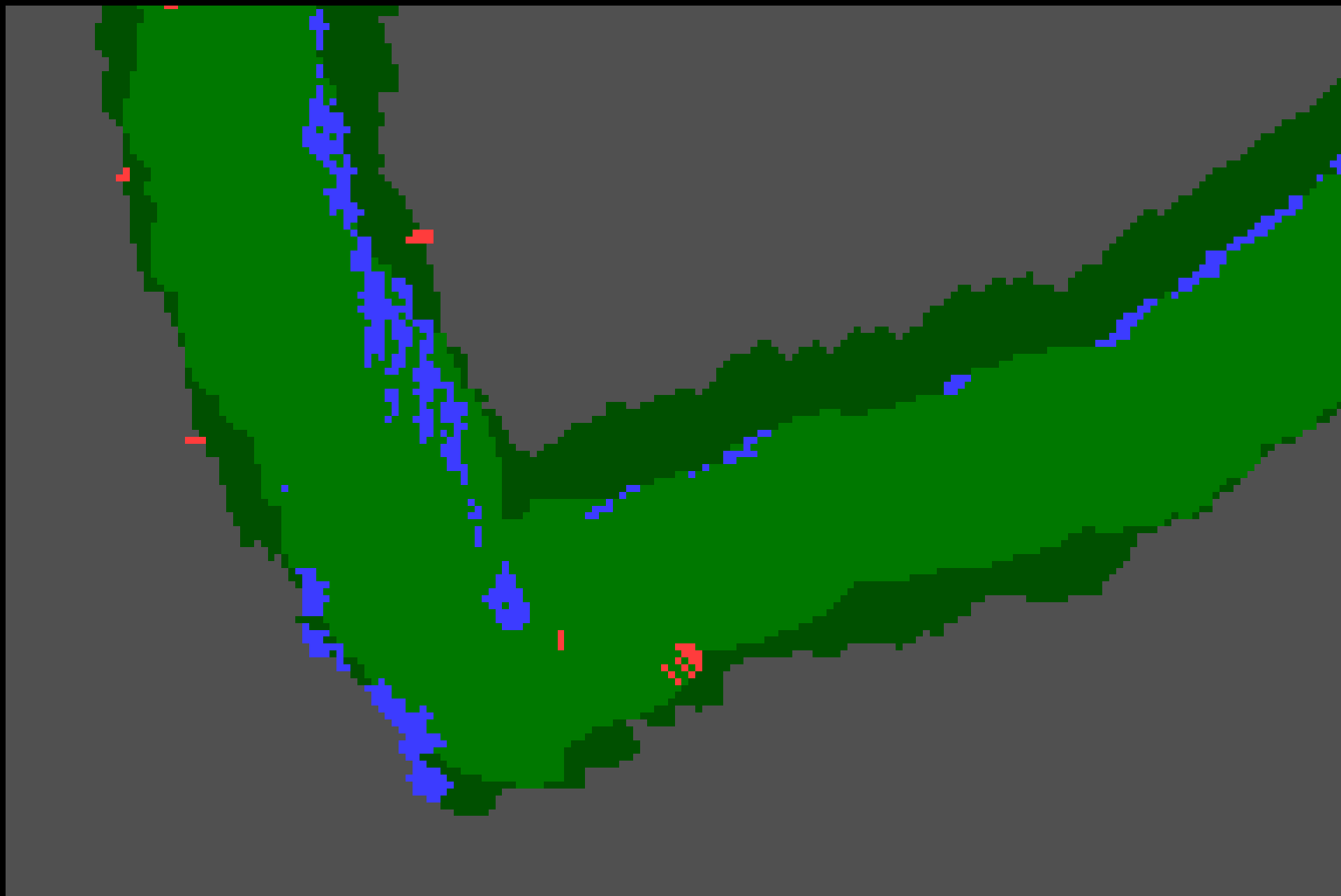
- Navigation points: 26 manually defined points
- GPS positions taken from OpenStreetMap
- Pickup and dropoff points in arrays (navig.cpp)
- Navigation path – defined by a string  
“N2N1 \*1 M4M5N7M3M8 \*2 M7M2M3N7M5N3”
- Compass calibration:
  - interval of 7 seconds - robot is moving straight
  - gps and compass azimuths to be fixed
  - performed every 30 seconds – 3 minutes



# WORLD MAP – BUILDING LOCAL GRID



- Local grid map
  - to store polar information from lidar and vision
  - 1 cell: 10 x 10 cm, 1 byte per cell, array[2000][2000]
  - always overwriting with new data - no heatmap
- Local position taken from GPS + odometry
  - wheel encoders provide speed information
- Colors used for visualization
  - Blue – grass (not-a-road) detected
  - Red – lidar obstacle
  - Green - no obstacle (light green = both sensors)



# USER INTERFACE

```
☐camera ☒vision ☒lidar ☒wmgrid reload

2017-09-16 16:29:09,176 INFO [process] istro::process_thread(): process_angle("MIN_MAX"):
navp_azimuth=999999.00, process_yaw=67.89, process_angle_min=0, process_angle_max=125,
process_angle=62, process_dir=90, yaw_src=1

2017-09-16 16:29:09,338 DEBUG [gps] istro::gps_writeData(): fix=1, latitude=49.213762,
longitude=18.743955, speed=0.262, course=47.85, gps_x=-13.11, gps_y=9.82, ref=1, navp_dist=-
1.000, navp_azimuth=999999.00, navp_maxdist=-1.000, navp_mindist=-1.000, navp_loadarea=-1,
lastp_dist=0.306, lastp_azimuth=52.64


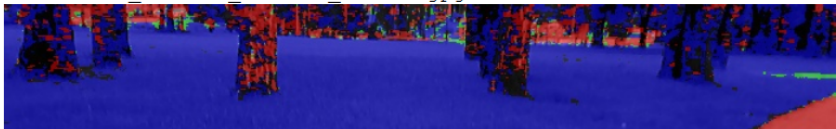
2017-09-16 16:29:09,175 DEBUG [process] istro::process_readData(): process_change=52,
gps_speed=0.401, gps_course=63.54, lastp_dist=-1.000, lastp_azimuth=90.00, ctrlb_ircv500=0.00,
ctrlb_angle=455, ctrlb_velocity=335, ctrlb_load=0, euler_x=90.62, ahrs_yaw=999999.00,
navp_dist=-1.000, navp_azimuth=999999.00, navp_maxdist=-1.000, navp_loadarea=-1, gps_ref=1,
gps_x=-13.36, gps_y=9.64

2017-09-16 16:29:08,554 INFO [process] istro::calib_process(): msg="calibration interrupted
(obstacle)!", calib.ok=0, process_angle_min=0, process_angle_max=125

2017-09-16 16:23:34,293 INFO [gps] istro::gps_thread(): msg="navigation point passed!", pos=6,
name="M3", navp_dist=10.382, navp_mindist=12.067

2017-09-16 16:23:34,293 INFO [gps] navig::navigation_next_point(): not found!, pos=6

2017-09-16 16:29:09,339 INFO [main] ControlBoard::write(): data="Dbye..."

out/rt2017_6539263_0015193_camera.jpg

out/rt2017_6539263_0015193_vision.jpg

```

- Visualisation in web browser works on notebook/tablet/phone
- Log parser is exporting interesting data do .json file
- Web page performs Ajax JSON requests every 1 second
- Images are only downloaded on demand (checkbox)

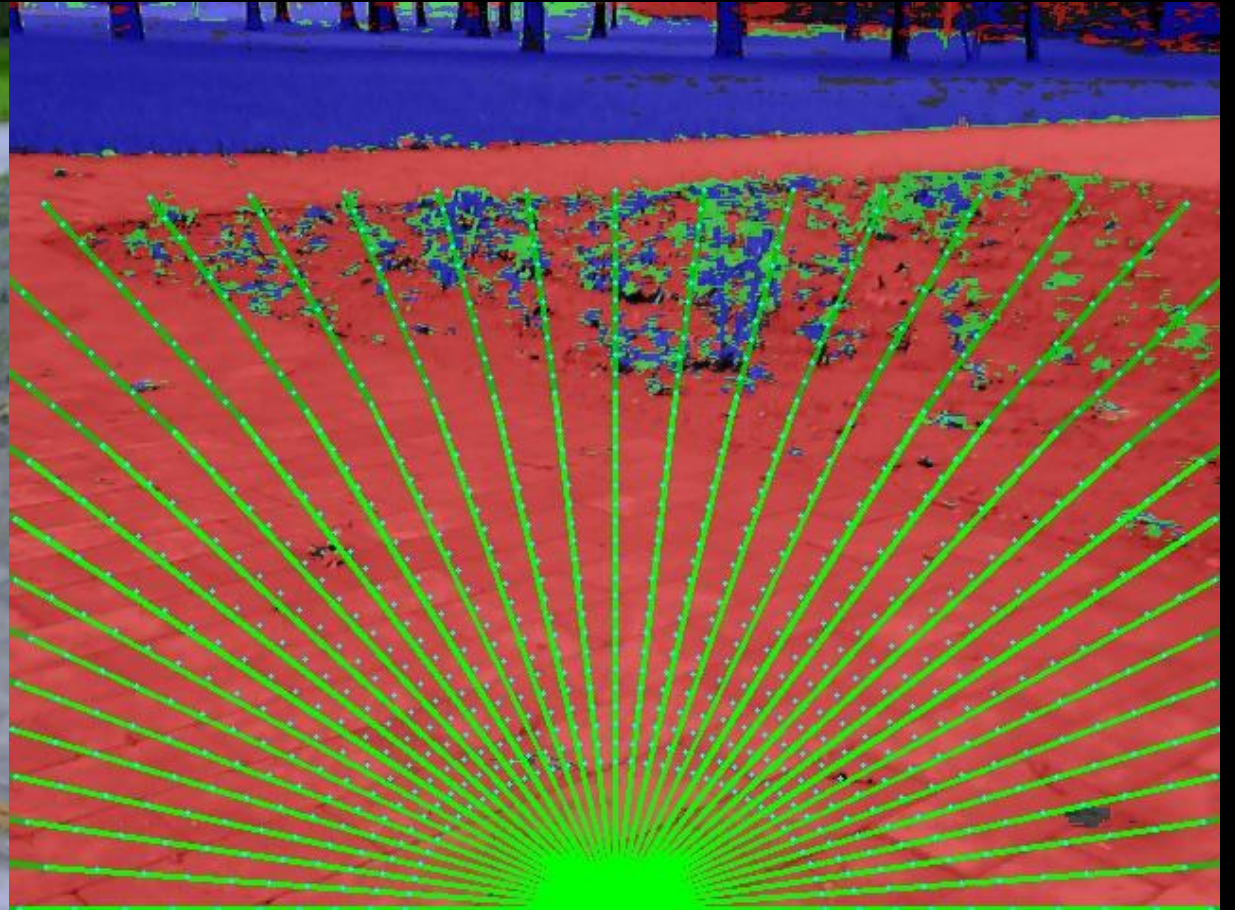


# PROBLEMS



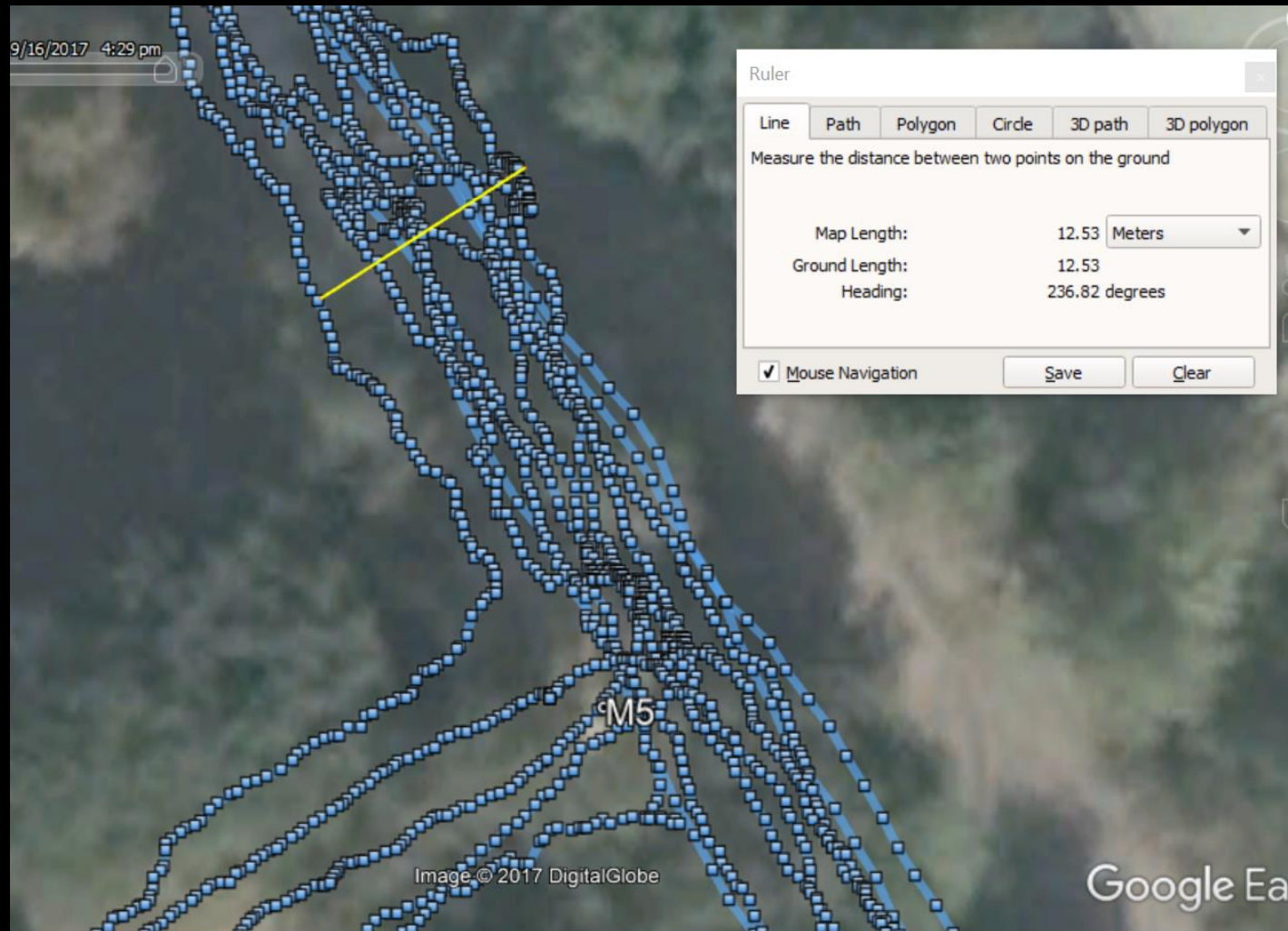
- Poor GPS performance (near service zone)
  - Incorrectly detecting pickup/dropoff points
  - impacted also compass calibration
  - => wrong calculation of navigation azimuth
- Local “traps” on roads (places for benches)
  - solution: tilt camera to look higher
- SW bug: sharp turns handled with “do nothing”
- Compass display did not work
- Robot Wifi AP failure (round2)
  - we were not able to enter GPS coordinates

# PROBLEMS – VISION FAILS DETECT ROAD





# PROBLEMS – POOR GPS PERFORMANCE





# PROBLEMS – SEGMENTATION FAULT

```
odroid@odroid: ~/projects/istro_rt2017
File Edit View Search Terminal Help
[New Thread 0xac2ff1a0 (LWP 6227)]
[New Thread 0xabaff1a0 (LWP 6228)]
[New Thread 0xa6f151a0 (LWP 6229)]

Thread 11 "istro_rt2017" received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0xabaff1a0 (LWP 6228)]
0x0003df02 in Lidar::drawOutput (this=0x61fc0 <lidar>,
    data=0x74f18 <dataset+76992>, data_cnt=@0x74f14: 311, img=..., stop=0,
    angle_min=0, angle_max=180, process_angle=96, process_angle_min=0,
    ---Type <return> to continue, or q <return> to quit---q
process_aQuit
(gdb) backtrace
#0  0x0003df02 in Lidar::drawOutput (this=0x61fc0 <lidar>,
    data=0x74f18 <dataset+76992>, data_cnt=@0x74f14: 311, img=..., stop=0,
    angle_min=0, angle_max=180, process_angle=96, process_angle_min=0,
    process_angle_max=162, image_number=11304)
    at /home/odroid/projects/istro_rt2017/lidar.cpp:668
#1  0x0002ac52 in save_thread (parg=0x0)
    at /home/odroid/projects/istro_rt2017/istro_rt2017.cpp:1960
#2  0xb6df95b4 in start_thread (arg=0x0) at pthread_create.c:335
#3  0xb6659bec in ?? () at ../sysdeps/unix/sysv/linux/arm/clone.S:89
    from /lib/arm-linux-gnueabi/libc.so.6
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
(gdb) █
```

● odroid@odroid: ~/projects/istro\_rt2017

File Edit View Search Terminal Help

#2 0xb6df95b4 in start\_thread (arg=0x0) at pthread\_create.c:335

#3 0xb6659bec in ?? () at ../sysdeps/unix/sysv/linux/arm/clone.S:89  
from /lib/arm-linux-gnueabi/libc.so.6

Backtrace stopped: previous frame identical to this frame (corrupt stack?)

(gdb) frame #0

Invalid character '#' in expression.

(gdb) frame 0

#0 0x0003df02 in Lidar::drawOutput (this=0x61fc0 <lidar>,  
data=0x74f18 <dataset+76992>, data\_cnt=@0x74f14: 311, img=..., stop=0,  
angle\_min=0, angle\_max=180, process\_angle=96, process\_angle\_min=0,  
process\_angle\_max=162, image\_number=11304)  
at /home/odroid/projects/istro\_rt2017/lidar.cpp:668

668                   img.at<Vec3b>(yn,xn)[0] = color2.val[0];

(gdb) print xn

\$1 = 44

(gdb) print yn

\$2 = 483

(gdb) print img

\$3 = (cv::Mat &) @0xabafeb04: {flags = 1124024336, dims = 2, rows = 480,  
cols = 640, data = 0xab21e010 '\024' <repeats 200 times>...,  
refcount = 0xab2ff010, datastart = 0xab21e010 '\024' <repeats 200 times>...,  
dataend = 0xab2ff010 "\001", datalimit = 0xab2ff010 "\001", allocator = 0x0,  
size = {p = 0xabafeb0c}, step = {p = 0xabafeb34, buf = {1920, 3}}}

(gdb)

# PRACTICAL EXAMPLES

...

# FREE SOURCE CODES



- Sources codes are available at GitHub as public project **Istro RT**:

<https://github.com/lnx-git/istro-rt>



THANK YOU

