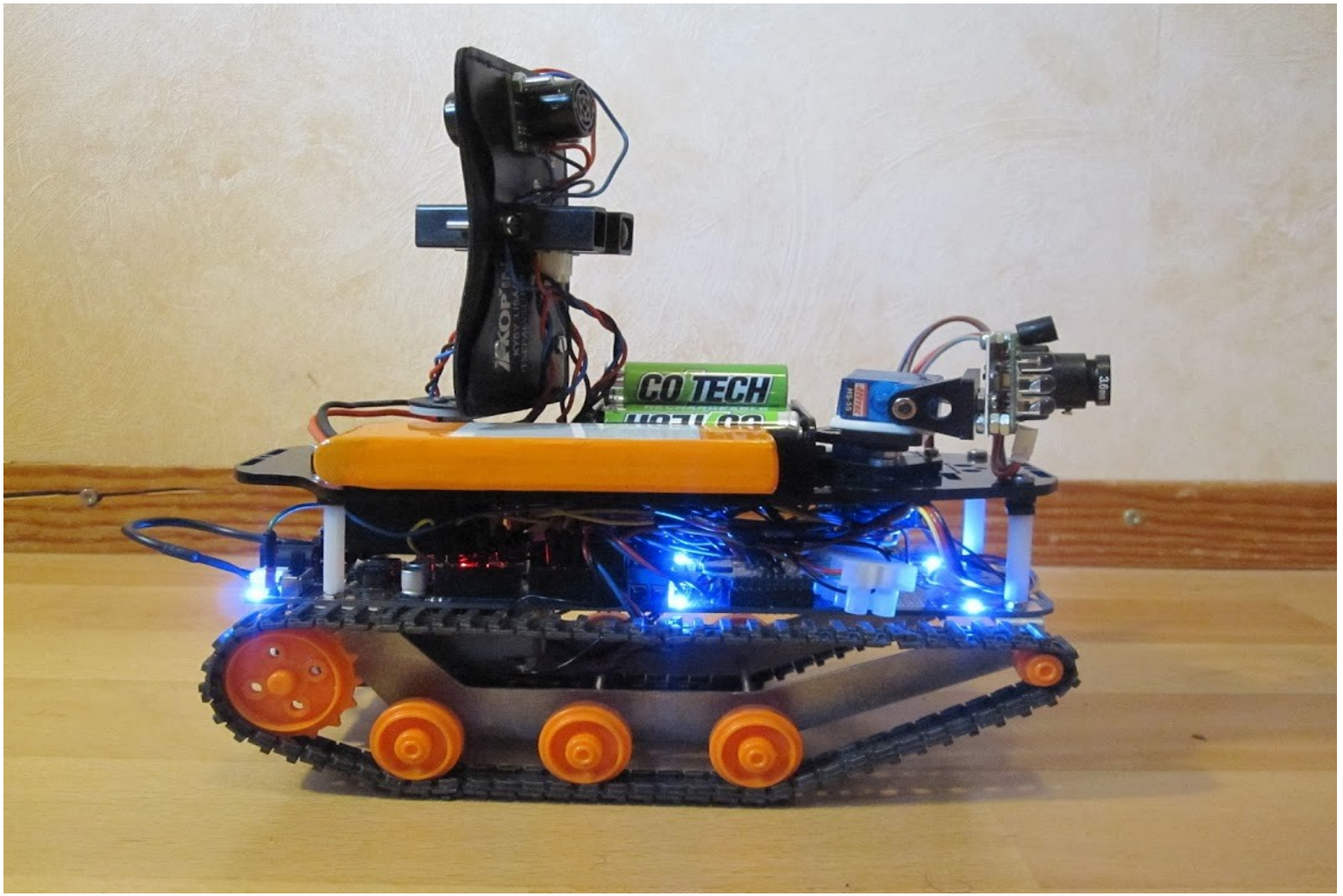
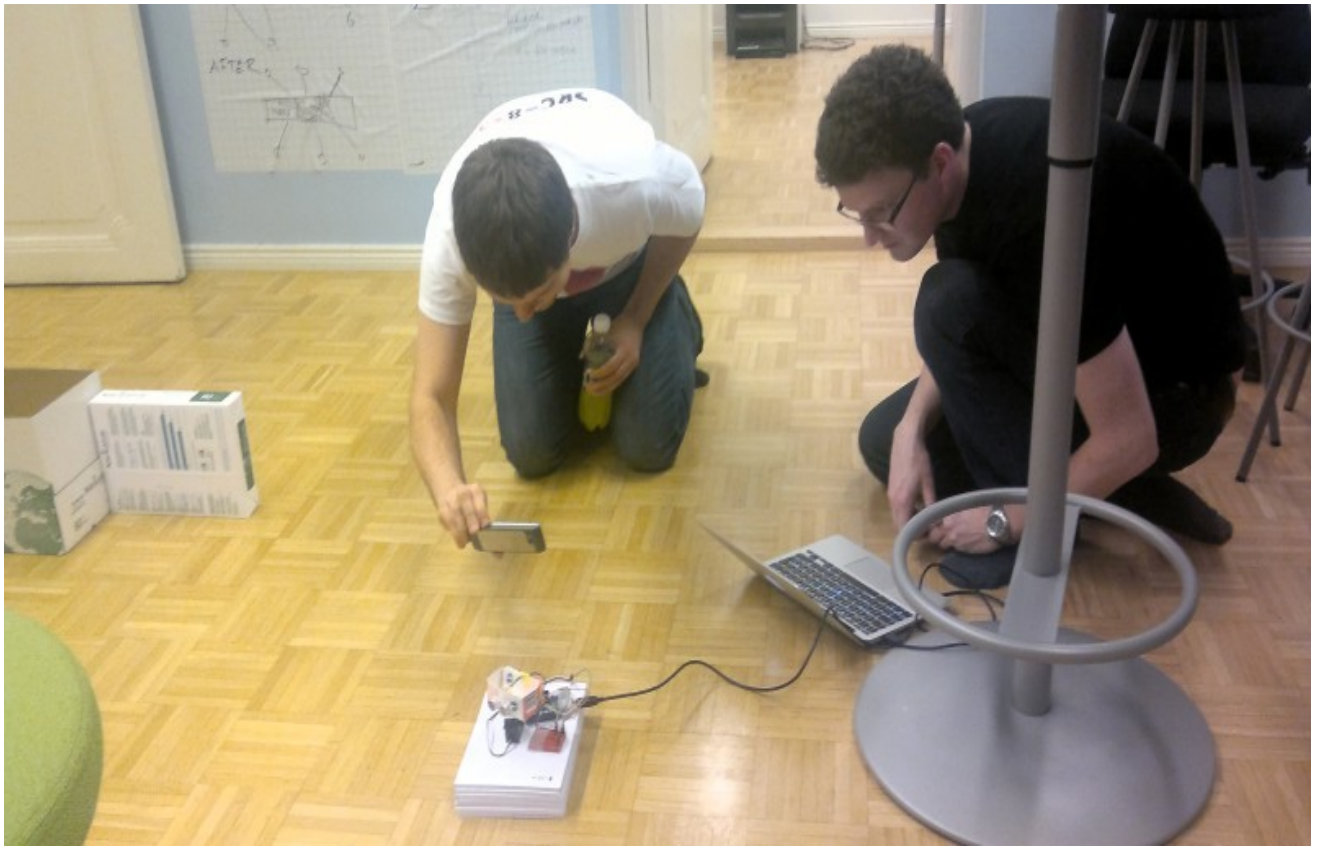
Story of RAISA



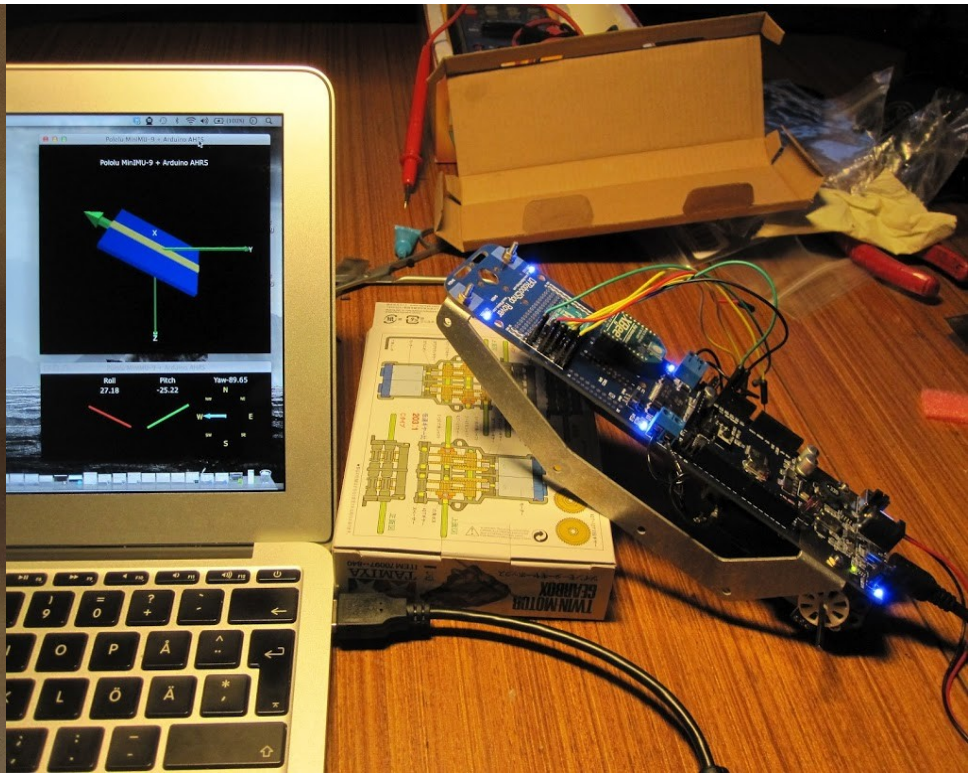
Background (1/2)

- Markku, Juha and Esa participated in online AI classes during 2011-2012
- Started experimenting with Arduino and sensors...

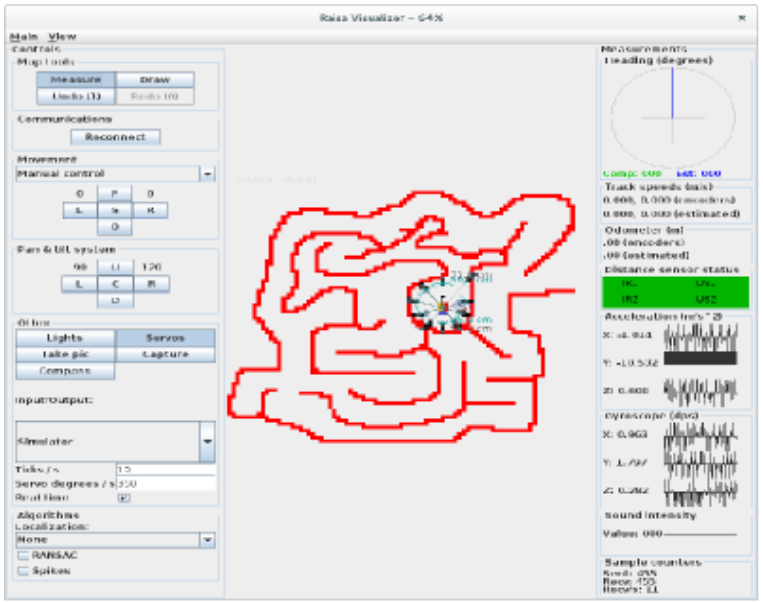


Background (2/2)

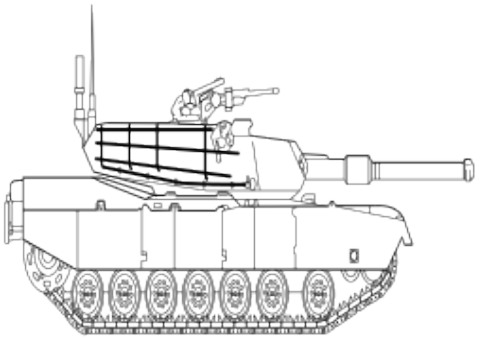
- Decided to build self-driving killer tank RAISA (=Robotti AI Suomen Armeijalle)
- Ordered first batch of parts from RobotShop on May 2012



Solution Architecture



XBee wireless



Raisa visualizer (Java)
- visualization
- manual controls
- simulator
- AI logic

Rover V2 XBee (Arduino)
- sensors
- actuators
- dummy drone

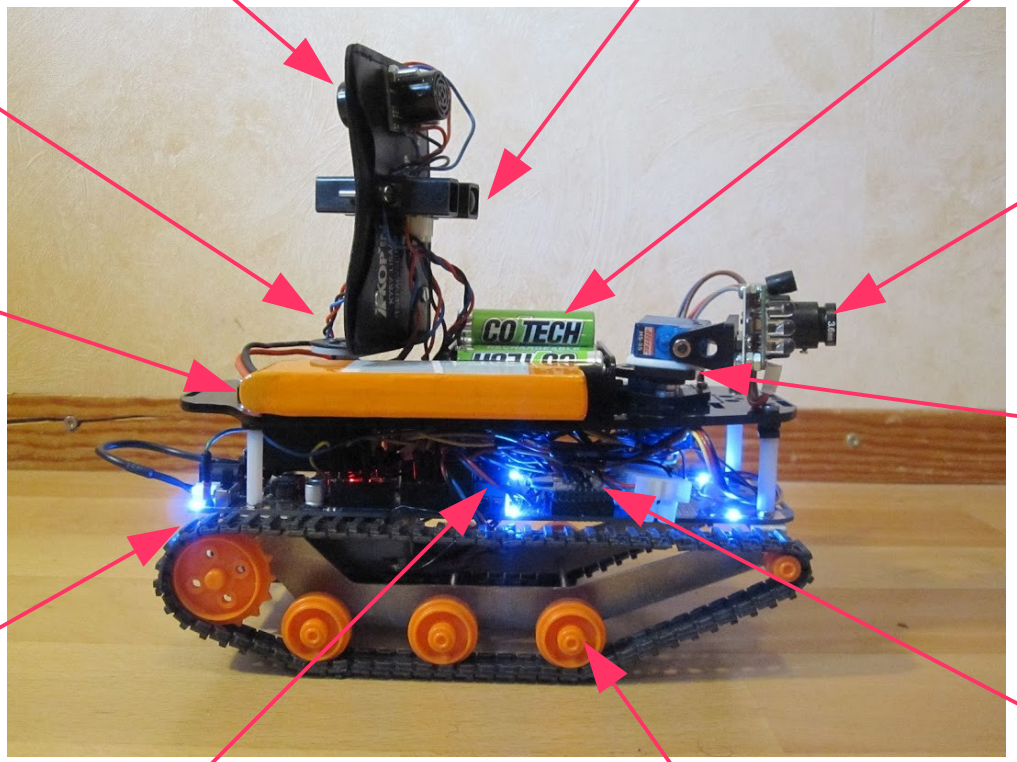
Tank Hardware

HS-322HD Standard Heavy Duty Servo

2 x Maxbotix LV-MaxSonar-EZ4 Sonar Module

2 x Sharp GP2Y0A02YK0F IR Range Sensor

Servos: 4 x 1.2V AAA battery pack



Color Serial JPEG Camera Module w / Infrared - TTL

DFRobot 7.4V Lipo 2200mAh Battery

ServoCity SPT50 Sub-Micro Pan Tilt Kit

Tamiya Twin-Motor Gear Box

Encoder Pair for Tamiya Twin Motor Gearbox

MinIMU-9: Gyro, Accelerometer, Magnometer, IMU

Maxstream 1mW XBee Transceiver Module

DFRobotShop Rover V2 - Arduino Compatible Tracked Robot

Tank Firmware

- Source code (~600 lines of C): https://github.com/Raisa/Raisa/blob/master/firmware/raisa/raisa.ino
- Loops the following algorithm infinitely:
  - Send rotate command to servo
    - Execute elapsed timers while waiting servo to turn
      - Poll and execute incoming commands every 20ms
      - Poll encoder ticks every 20ms
  - Read sensor statuses
  - Send sensor statuses to Visualizer
  - If take picture-command pending, stop and take picture

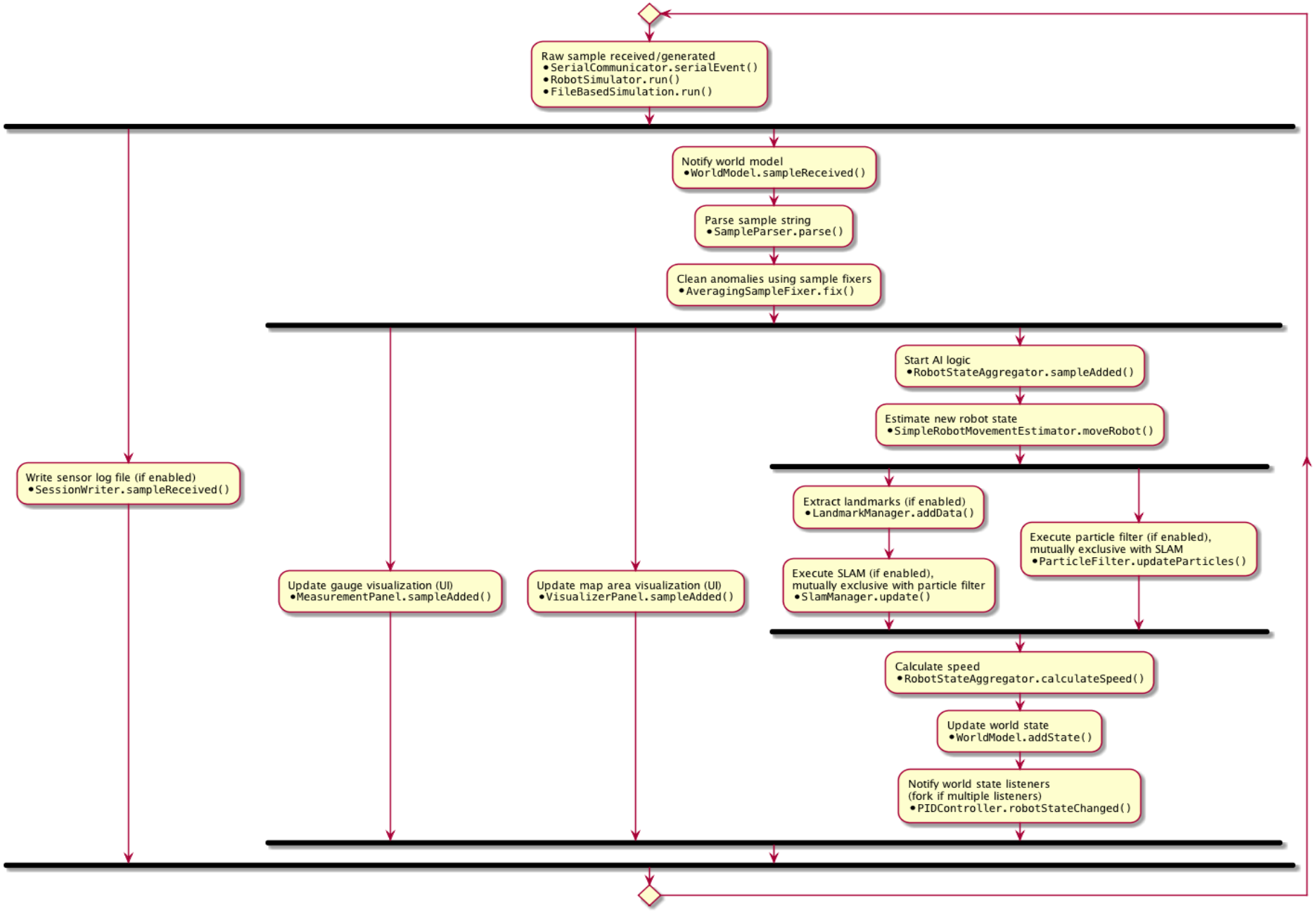
Raisa Visualizer (1/3)

- Java Swing application
  - Requires patched native library librxtx for serial communication
  - Serial communications does not work on Linux, because custom 111111 baud rate is not supported by unpatched librxtx
    https://github.com/Raisa/Raisa/wiki/XBee-configuration
- Source code (~10000 raw lines)
  https://github.com/Raisa/Raisa/tree/master/raisavis
- Controls (left bar)
  - Map tools (measure, draw, undo, redo)
  - Movement (manual through keyboard or PS3 SixAxis, PID, A* route planning)
  - Pan & tilt for camera
  - Others (lights, servos on/off, take pic, capture control and sensor data, compass on/off, input/output source)
  - Algorithms
    - Localization (particle filter, SLAM)
    - Landmark extraction (RANSAC, Spikes)

Raisa Visualizer (2/3)

- Map area visualization (middle area)
  - Map
  - Robot position and trail
  - Distance sensor data
  - Landmarks
  - Algorithm specific stuff (e.g. particles)
- Measurements (right bar)
  - Heading (based on encoders and compass)
  - Drive chain speeds
  - Odometer
  - Other IMU data (acceleration, gyro)
  - Sound sensors
  - Sample counters

Raisa Visualizer (3/3)



Lessons Learned: Hardware (1/3)

- IR and ultrasonic sensors are insufficient for advanced AI applications
  - Maximum of 40 distance measurements per second is not enough for e.g. SLAM. LIDAR would give 2000 accurate measurements per second.
  - Lots of other hardware seems also to cause electrical interference making measurements inaccurate
- Sound sensors are useless
  - Low data transmission speed (111111 bauds) prevents transferring audio
- IMU is useless
  - Low speed makes accelerometer useless
  - Flat operating environment makes gyro useless
  - Electrical interference makes compass useless

Lessons Learned: Hardware (2/3)

- Wireless Xbee is too slow
  - Default XBee is 9600 baud. It can be tweaked to 111111 bauds (Arduino 16MHz makes 115200 baud unreliable), which is still not enough
- Serial camera is too slow
  - Arduino can read picture from the camera over serial connection in chunks, which are resent over XBee serial using. This is done sequentially using single Arduino thread. Capturing even one single resolution picture takes many seconds.
- Arduino problems
  - Multiple threads would allow reading input/output and sensors (e.g. camera) simultaneously
  - Jumper wires pop out easily: difficult to spot the problem with dozens of cables (soldered connections needed)
  - Side note: WIFI shield tested in another project. Finding: difficult to program and unreliable

Lessons Learned: Hardware (3/3)

- Logic Analyzer is very useful tool for debugging hardware problems
- Bottom line: use less components of more quality

Lessons Learned: Visualizer

- Difficult to make simulator and real hardware match
  - Sensor noise undershooting or overshooting
  - Hardware is not ideal
    - Motors on different wheels have different power
    - Center of mass is not in the middle of vehicle
- Document coordinate system if not homogenous throughout the project (radians or degrees, which direction the axes point, which direction is positive angle, etc)
- Difficult to divide into switchable components: different algorithms that calculate same output can take different input

“Raisa 2”

- Use more advanced components
- Rewrite Visualizer (some code can be cut & pasted from Raisa 1)
- Write drone software from scratch using something else than Arduino
- Still differential drive (=two wheel/chain) because of easier physics

“Raisa 2” Hardware Proposal

- RPLIDAR 360° Laser Scanner (360e)
  http://www.robotshop.com/eu/en/rplidar-360-laser-scanner.html
- Arduino Tracked Mobile Tank Robot Kit (650e)
  http://www.robotshop.com/eu/en/arduino-tracked-mobile-tank-robot-kit.html
- Raspberry Pi Model B Computer Board w/ 8GB SD Card (41e)
  http://www.robotshop.com/eu/en/raspberry-pi-model-b-computer-board-8gb-sd-noobs.html
- Mini USB WiFi Module (11e)
  http://www.robotshop.com/eu/en/mini-usb-wifi-module.html
- USB Battery Pack - 6600 mAh (27e)
  http://www.robotshop.com/eu/en/usb-battery-pack-6600-mah.html