TECHNICAL UNIVERSITY OF CRETE
PRODUCTION ENGINEERING AND MANAGEMENT

HARDWARE AND SOFTWARE UPGRADE OF THE ATRV-mini OUTDOOR VEHICLE

Master Thesis

Ntegiannakis Theodosis

Supervisor: Nikos Tsourveloudis

Chania, August 2017

Table of Contents

Table of Contents 3
Chapter 1 Introduction 5
1.1 Autonomous robotic vehicles 5
1.2 Surveillance 5
1.3 Scope 5
1.4 ATRV-mini 7
Chapter 2 Hardware Upgrade Process 9
2.1 Controller 10
2.2 IMU 11
2.3 GPS 12
2.4 Ultrasonic Sensor 13
2.5 Camera 14
2.6 Motor driver 15
2.7 Batteries and Solar Panel 15
2.8 Connection 17
2.9 Upgraded ATRV-mini 18
Chapter 3 Software Upgrade 19
3.1 About ROS 19
3.2 ROS Concepts 19
3.2.1 Nodes 19
3.2.2 ROS Master 19
3.2.3 Messages 20
3.2.4 Topics 20
3.2.5 Bags 20
3.2.6 Packages 20
3.3 ROS Standard Messages 21
3.3.1 REP 103 Standard Units of Measure and Coordinate Conventions 21
3.3.2 REP 105 Coordinate Frames for Mobile Platforms 22
3.4 ROS Package Development 23
3.4.1 /imu 23
3.4.2 GPS 23

3.4.3 Camera 23
3.4.4 /analogread 24
3.4.5 /position 24
3.4.6 /wheel_controller 24
3.4.7 /move_to_point 25
3.4.8 /path_planner 25
3.4.9 /follow_object 26
Chapter 4 Experiments 29
4.1 Remote Control 29
4.2 Move to point 29
4.3 Waypoint Navigation 30
4.4 Follow object 30
4.5 Energy consumption 32
Chapter 5 Conclusion 34
References 35

Chapter 1 Introduction

1.1 Autonomous robotic vehicles

Autonomous surveillance robot control is a complex problem of major interest at the research community. Systems capable of performing efficient and robust autonomous navigation are unquestionably useful in many robotic applications [1] such as manufacturing technologies [2], urban transportation [3], assistance to disabled or elderly people [4] and surveillance [5].

1.2 Surveillance

The use of robotics in surveillance and inspection is not a novel topic, as different kinds of robots have been used in various applications. In France an unmanned aerial vehicle (UAV) has been used for structure monitoring and bridge maintenance [6], in Korea a robotic system has been used for inspecting the safety status of bridges [7]. Another example is the pipe inspection robot for magnetic crack detection of iron pipes [8].

The key benefits of using surveillance robots instead of a static sensor network include [9]:

- High versatility, as, depending on the application, different sensors can be attached to the robots.
- Robots provide various viewing angles that would otherwise be difficult to cover with a sensor network.
- Can be deployed in areas where hazardous materials can be present.

Easy to upgrade. Sensor technology is changing and upgrading a wireless sensor network would require changing a relative large number of sensors.

1.3 Scope

This project deals with the upgrading of a Real World Interface ATRV-mini all-terrain robotic vehicle. The Intelligent Systems and Robotics Laboratory [10] purchased two ATRV-mini robots in 2000. The robots were fully operational for more than 4 years [11] [12] [13] [14] taking part in several research and educational projects.

Two main disadvantages were highlighted very early with the experience using this robot:

- Its energy reserves were inadequate for its operational needs. Its battery life was less than the 3 hours period specified by the manufacturer.
- A part of robot's controller was based on portable computer components. Thus the, mandatory for that era, use of a hard disk drive as robot's storage device made it prone to vibrations present during outdoors use. The result was frequent unexpected program terminations.

Seventeen years after its first usage the robot is now not operational due to the following reasons:

- Its electronics equipment had reached their end of life

- Developing, upgrading or updating its software is prohibited by its outdated hardware limitations

However its frame, wheels, electric motors and transmission are fully functional.

In this project the upgrading of the outdated electronics and usage of up to date software will take place so that the robot will be fully functional again and ready to be used again for the purposes of the Laboratory.

1.4 ATRV-mini

The ATRV-mini is an all-terrain robotic vehicle that turns by using skid steering. It is an autonomous, agile robotic vehicle, suitable for all terrain types, that inherits all the advantages of other larger ATRV robots, plus the unique advantages of smaller size and reduced weight.

Figure 1 ATRV-mini vehicle in its original form

The ATRV-mini specifications as found in the manufacturer's manual are shown in the table below.

Table 1 ATRV-mini specifications

Dimensions	62x53x45 cm (LxWxH)
Clearance	7.62 cm
Weight	38.6 kg
Body	Formed and welded aluminum
Speed	0 – 1.5 m/sec
Run Time	3 to 6 hours, terrain dependent
Drive	4 – wheel
Turn Radius	Zero (turns on center)
Tires	25.4 mm (10 in.) pneumatic knobby
Batteries	Two 12V, 12 Ah
Motion Control	RWI rFLEX System
Motors	Two 0.10 HP, 24V DC
Computer	Pentium III EBX
Software	RWI Mobility Robot Infrastructure
I/O Ports	Ethernet, Rs-232, Joystick
Sensors	Sonar: 24 (6 each front corner, 6 each rear corner) Camera (Sony EVI D30) Global Position System (GPS) receiver Inertial Measurement Unit (IMU) Wireless communication

The ATRV-mini was originally equipped with a RedHat Linux powered Pentium III computer system. Its main disadvantage was the frequent unexpected program terminations due to the use of a hard disk drive unit which was prone to the heavy vibrations being present during outdoors use.

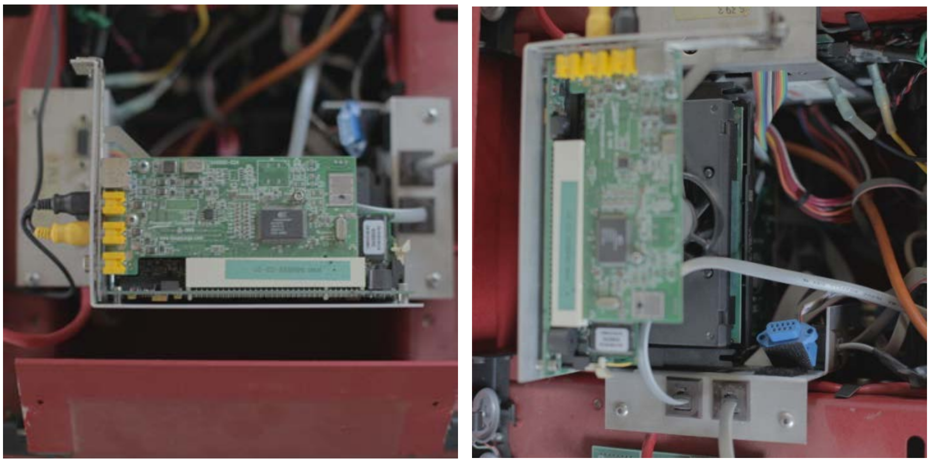


Figure 2 ATRV-mini Pentium III computer system

Obstacle avoidance was accomplished using 24 ultrasonic sensors, 6 on each of its four corners. A dedicated ultrasonic controller was placed to control the 6 sensors on each corner.

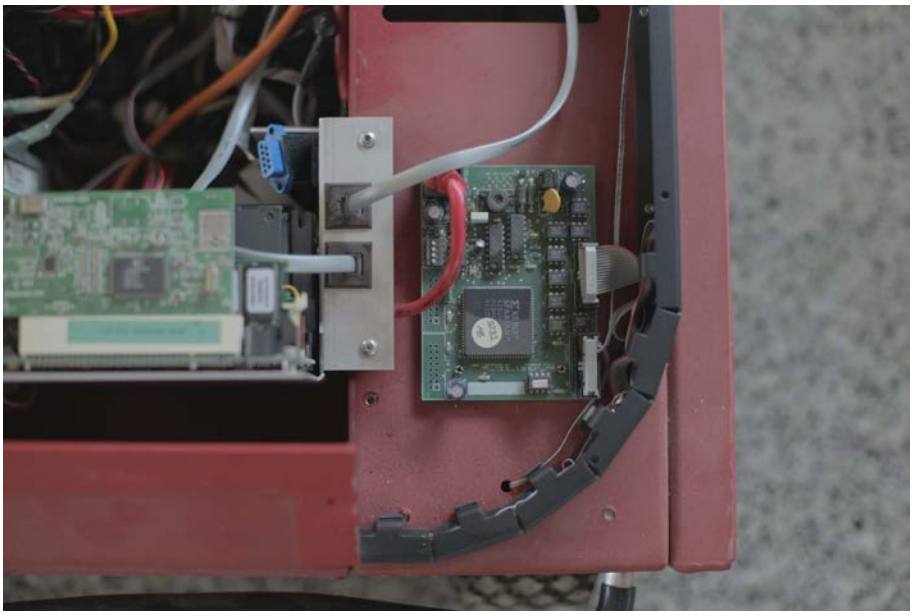


Figure 3 ATRV-mini ultrasonic sensor controller next to the ultrasonic sensors

Chapter 2 Hardware Upgrade Process

During the reconstruction mechanical parts such as the robot's metal frame, wheels, motors and transmission system where remained intact. In the contrary, robot's electronic components were replaced with up to date, off the shelf components. All the new components used are shown in the following paragraphs.

Figure 4 ATRV-mini with the electronics removed

2.1 Controller

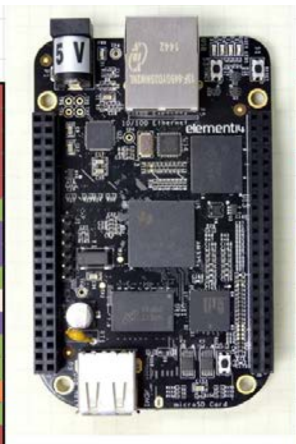
The robot's original electronics configuration was based on a Linux [15] powered Pentium III [16] computer system. In addition, the robot was equipped with rFLEX system for motion control and diagnostics. The rFLEX system was removed while the Pentium III system was replaced with a BeagleBone Black (BBB) single board computer [17]. BBB is a low-cost credit-card-sized development platform with active community support. It features an ARM Cortex-A8 1Ghz processor, 512MB DDR3 RAM, 4Gb on board eMMC flash storage with Ethernet and USB support. It is capable of running Linux operating system (such as Debian, Ubuntu, etc).

Figure 5 BeagleBone Black

Apart from the above the Beaglebone Black board was selected because of its Input and Output ports. It has 2x46 pin headers providing more than enough GPIO, PWM, Analog, Serial ports for the project.

P9

Function	Physical Pins		Function
DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V
VDD 5V	5	6	VDD 5V
SYS 5V	7	8	SYS 5V
PWR_BUT	9	10	SYS_RESET
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPIO_CSO	17	18	SPIO_D1
I2C2_SCL	19	20	I2C_SDA
SPIO_DO	21	22	SPIO_SLCK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SP11_CSO
SP11_DO	29	30	GPIO_112
SP11_SCLK	31	32	VDD_ADC
AIN4	33	34	GND_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPWMO
DGND	43	44	DGND
DGND	45	46	DGND



LEGEND

LEGEND
Power, Ground, Reset
Digital Pins
PWM Output
1.8 Volt Analog Inputs
Shared I2C Bus
Reconfigurable Digital

P8

Function	Physical Pins		Function
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DATO	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

Figure 6 Beaglebone Black Pins

2.2 IMU

IMU stands for Inertial Measurement Unit. An IMU depending on how many depth of freedom (DOF) is capable of, is usually equipped with an accelerometer measuring linear acceleration on 3 axis, a gyroscope measuring angular velocities in 3 axis, and a magnetometer measuring earth's magnetic field in 3 axis.

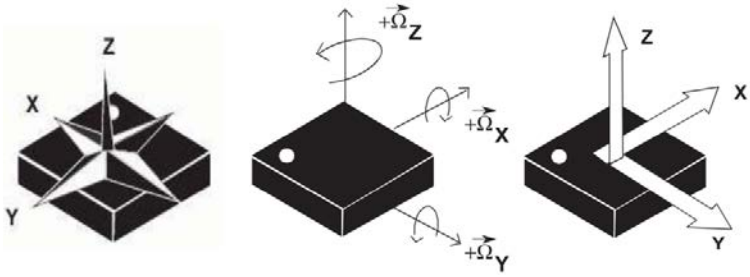


Figure 7 9 DOF IMU axis

ATRV-mini was equipped with an electronic compass sensor. It was replaced with a Sparkfun 9DOF Razor IMU M0 [18]. In addition to the accelerometer, magnetometer and gyroscope it provides an SAMD21 32bit, 48Mhz Arduino compatible microprocessor and a microSD card. It is a very powerful board that communicates via USB or serial and due to the existence of a microprocessor it is capable of making all the needed calculations relieving the robot's computer from them.

Figure 8 Razor IMU M0

2.3 GPS

The Global Positioning System (GPS) is a space-based navigation system. It has brought a breakthrough in navigation since its first appearance and now it is a standard in ships, airplanes, cars and mobile devices. A GPS receiver communicates with other devices by sending defined messages called NMEA [19] which contain information such as longitude, latitude, altitude, velocity, heading angle etc.

In RWI's documentation there were no detailed specifications for the GPS originally installed. The GPS module that was selected is an Adafruit Ultimate GPS Breakout Board [20] capable of tracking 22 satellites, 10Hz update rate, NMEA 0183 and position accuracy of 1.8m.

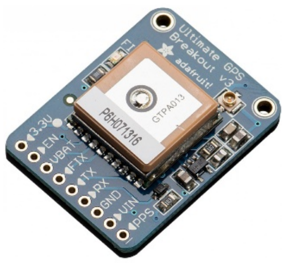


Figure 9 GPS Breakout Board

The GPS module is connected with an external antenna as shown in the figure bellow.

Figure 10 GPS With Antenna

2.4 Ultrasonic Sensor

ATRV-mini in its original configuration was equipped with 24 ultrasonic sonars. In each one of its four corners there were six sonars with a dedicated controller. The original sonars including their controllers were replaced with four ultrasonic range finders, three at the front part and one at the rear. The sensor used is MaxBotix LV-MaxSonar EZ1 MB 1010 [21] capable of 1 inch resolution, 20Hz reading rate.

Figure 11 MaxSonar EZ1

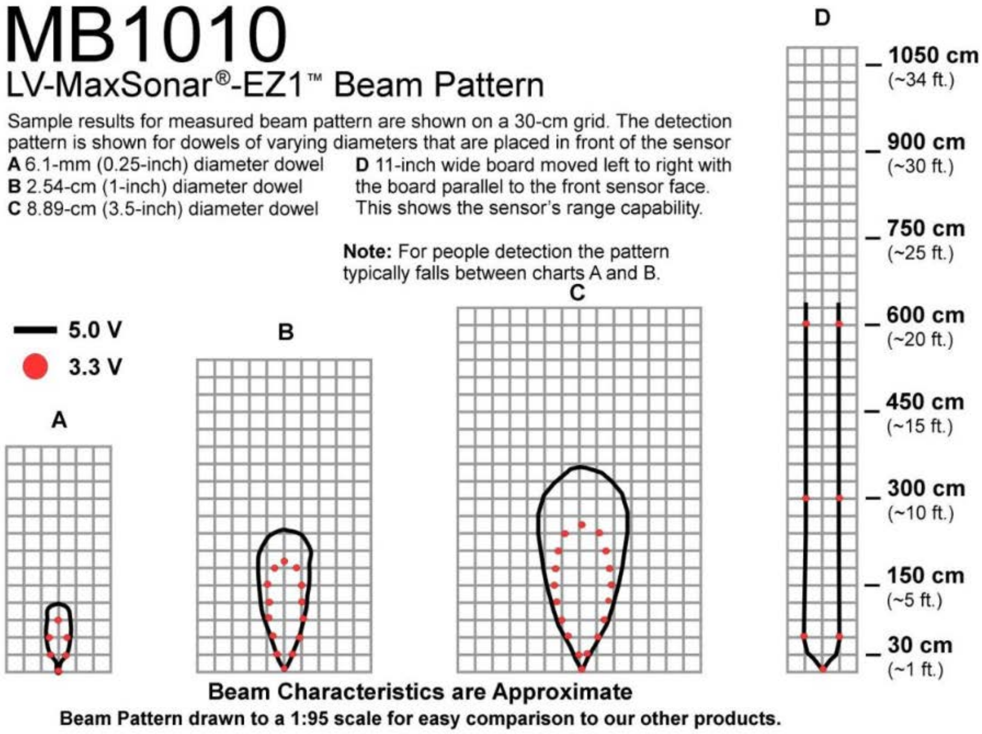


Figure 12 Ultrasonic Beam Pattern

Three sensors are placed in the front of the robot and one on the back.

2.5 Camera

ATRV-mini sensors' suite included a 0.38MP (Mega Pixel) Sony EVI D30 color pan-tilt-zoom video camera including an automatic target tracking and motion detector. Replacing the previous camera with a 1 MP is a noticeable upgrade. But the Pixy CMUcam5 [22] that was selected is more than just a 1MP camera. It is a low cost, low power, open source camera with an embedded 204 Mhz dual core processor. This enables the camera to execute image recognition algorithms with its onboard processor relieving the main robot's computer from this very CPU intensive work.

Figure 13 Pixy CMUCam

Pixy CMUcam5 can save up to 7 different colors and track hundreds of objects at the same time at 50 frames per second.

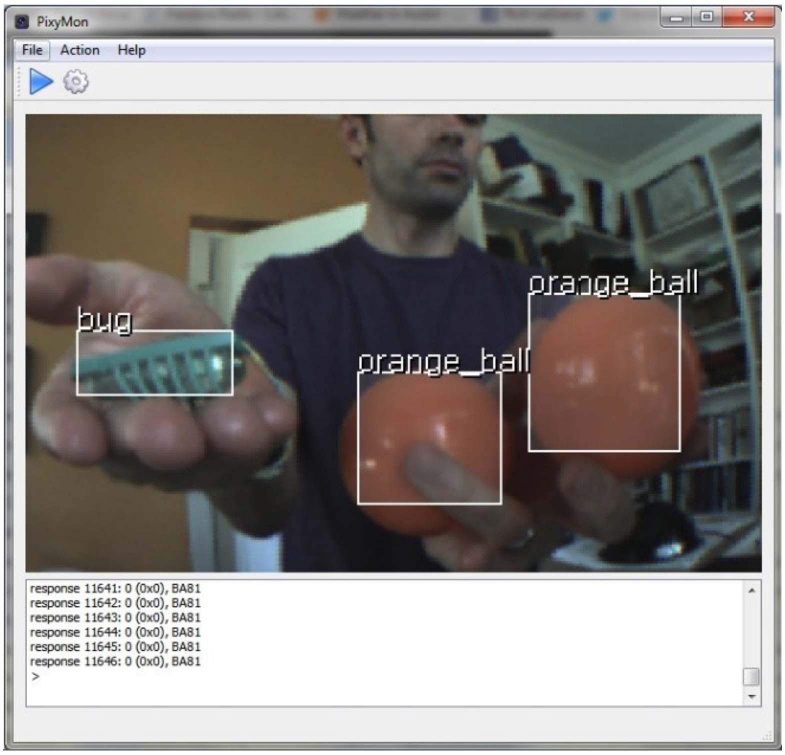


Figure 14 Object recognition CMUcam

2.6 Motor driver

For the driving of the two DC motors of the robot a new motor driver is selected. The Sabertooth 2x12 produced by Dimension Engineering [23] is a motor controller able to drive two DC motors at 24V at 12A continuous current. Controlling the motors is made either by analog voltage, radio control and serial. For the purposes of this project the communication with the Beaglebone Black is made by serial connection.

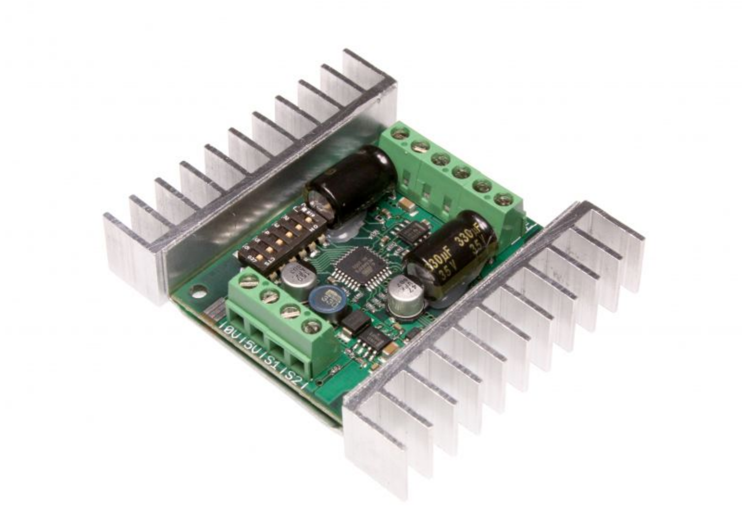


Figure 15 Sabertooth 2x12 Motor Driver

2.7 Batteries and Solar Panel

The original 2x12V 12Ah lead acid batteries were replaced by 2x12V 7Ah ones succeeding a total weight reduction of 2kg. To confront with the small autonomy time a solar charging system selected to equip the ATRV. Solar panel used are three Sparkfun 5.2W panel [24] with dimensions 180x220mm.

This unit is rated for 8V open voltage and 650mA short circuit. After measuring a unit outside resulted in 9.55V open voltage and 550mA short circuit

Figure 16 Solar Panel

The following figure shows the removed electronics placed next to the new ones used in this project. The difference in size of the modern electronics used is obvious when compared to the seventeen years old originally used in the robot.

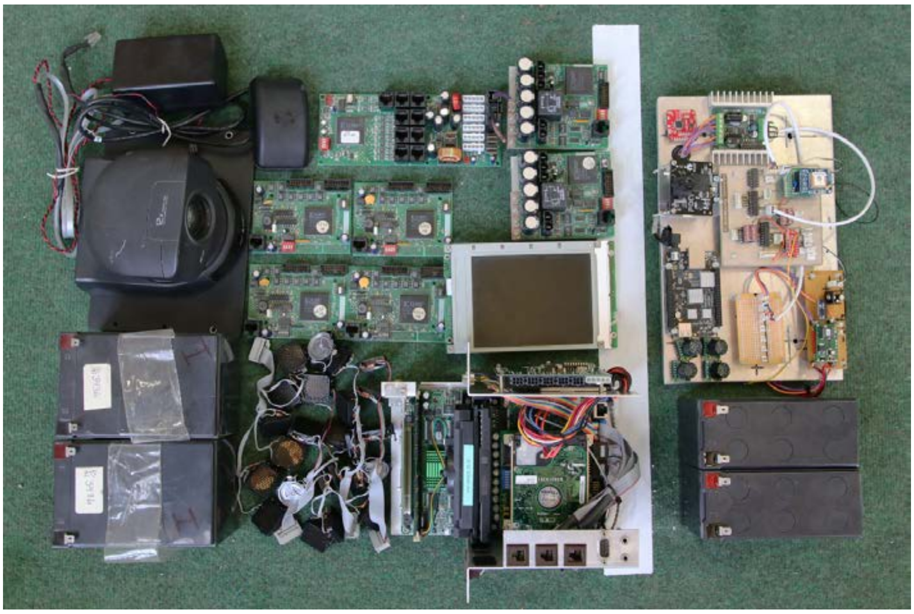


Figure 17 Removed electronics (left side of the line) next to the new ones used (right side)

2.8 Connection

The connection of all the components used is shown in the figure below. Beaglebone Black is receiving data from all sensors and is sending commands to the motor driver to control the motors.

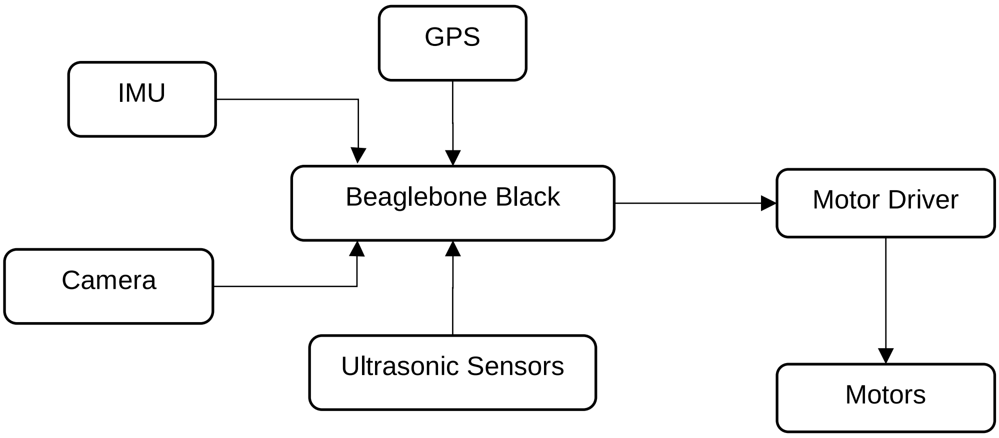


Figure 18 Wiring of the components

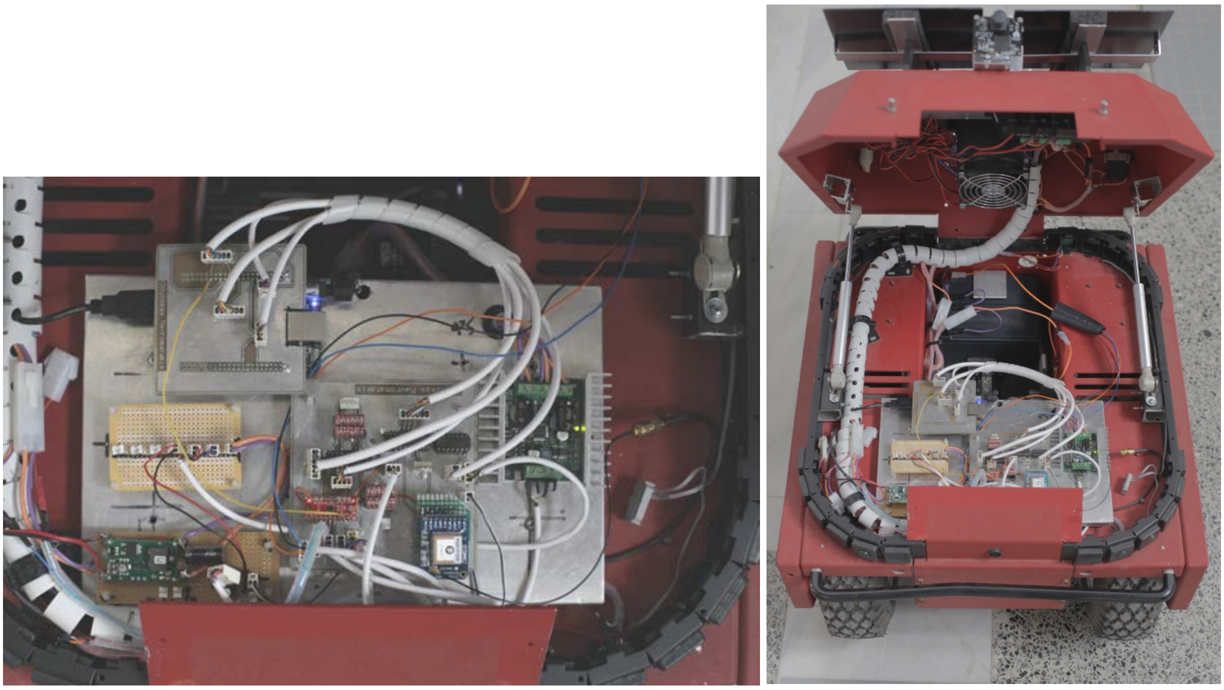


Figure 19 Electronic components placed inside the ATRV-mini

.

2.9 Upgraded ATRV-mini

After the electronics upgrade the ATRV-mini is ready for software development and indoors / outdoors testing.

Figure 20 Exterior of ATRV-mini with all new components installed

Chapter 3 Software Upgrade

RWI provided a software module called Mobility as an Application Protocol Interface (API) for the development of custom robotic controllers. Mobility was an object-oriented framework consisting of software tools for programming in Java or C++ and accessing robot's devices such as motors and sensors. Code development was simplified as Mobility was a successful software layer between the user applications and the robot. Mobility system was replaced in this project with a Robotic Operating System (ROS) [25]

3.1 About ROS

ROS is an open-source, meta-operating system for robots. It consists of tools, libraries and conventions that aid the developer in the creation of complex robotic behavior that would be very hard to be developed by a single individual, laboratory or institution.

Figure 21 ROS Logo

There are similar robot frameworks, such as Player [26], Orocos [27], Carmen [28], Orca [29], MOOS [30] and Microsoft Robotics Studio.

ROS is open source (using the permissive BSD open source license) and is based on collaborative software development so that groups can use each other's work in their projects. Now, ROS ecosystem consists of tens of thousands of users worldwide.

ROS makes it easier to take advantage of a distributed computing environment.

3.2 ROS Concepts

ROS is a powerful and flexible tool but it is also very complex. This chapter will be a description of the basic concepts of ROS that were used in this project.

3.2.1 Nodes

Nodes are the processes that perform computation. They are part of the Computational Graph, ROS peer to peer network that are processing data together.

3.2.2 ROS Master

ROS Master provides name registration and lookup to the rest of the Computational Graph. With the presence of ROS Master, nodes are able to find each other and to communicate.

3.2.3 Messages

Messages is the way that nodes communicate with each other. Message is a simple data structure comprised of typed fields. Primary data types (such as Integers, Boolean, Floats, Characters, etc.) are supported as well as arrays of those. Messages can also include other messages.

3.2.4 Topics

Messages are routed through ROS via a transport system with publish / subscribe semantics. A node sends a message by publishing it to a topic. A topic's name is used to identify the information is published on it. Likewise, a node that is interested on a certain kind of data, subscribes to a topic and immediately gains access to the messages published in that topic.

Multiple nodes can subscribe and publish on one topic but only one type of message can be used. One node can also publish or subscribe on multiple topics.

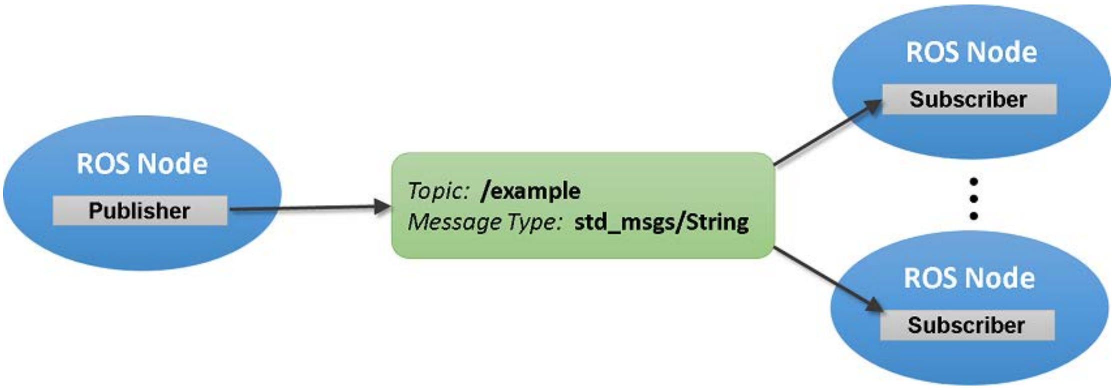


Figure 22 ROS Nodes Topic Example

3.2.5 Bags

Bags are a format for saving and playing back ROS message data. Bags are useful for storing data such as sensor data and use them for developing and testing algorithms.

3.2.6 Packages

Packages are part of the ROS filesystem. They are the main unit for organizing software in ROS. Inside a package there may be nodes, libraries, datasets, configuration files and everything else that is useful.

For the purposes of this project a package will be created, containing all the nodes, messages and configurations needed.

3.3 ROS Standard Messages

ROS is using ROS Enhancement Proposals known as REPs. Two important REPs used for the implementation of the ROS framework are REP 103 [31] and REP 105 [32].

3.3.1 REP 103 Standard Units of Measure and Coordinate Conventions

REP 103 [31] provides a reference for the units and coordinate conventions used within ROS.

ROS is using SI standardized units. The units which are commonly used are:

Quantity	**Unit**
Base Units	
length	meter
mass	kilogram
time	second
current	ampere
Derived Units	
angle	radian
frequency	hertz
force	newton
power	watt
voltage	volt
temperature	celsius
magnetism	tesla

All systems are right handed. This means they comply with the right hand rule.

In relation to a body the standard is:

```
X forward
Y left
Z up
```

For short-range Cartesian representations of geographic locations, use the east north up (ENU) convention:

```
X east
Y north
Z up
```

For outdoor systems where it is desirable to work under the north east down (NED) convention, an appropriately transformed secondary frame is defined with the "_ned" suffix:

```
X north
Y east
Z down
```

3.3.2 REP 105 Coordinate Frames for Mobile Platforms

REP 105 [32] specifies naming conventions and semantic meaning for coordinate frames of mobile platforms used with ROS.

REP 105 defines the base_link, map, odom and earth coordinate frames, the relationship between them and how multi-robot are represented in the earth coordinate frame.

“earth” coordinate frame is the origin of ECEF (Earth Centered – Earth Fixed) [33] in which multiple map frames of multiple robots are interacted.

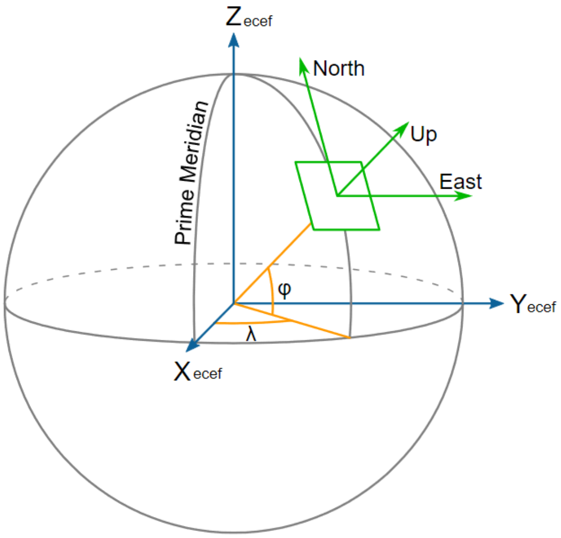


Figure 23 ECEF with a tangential map frame

3.4 ROS Package Development

Multiple ROS nodes were developed as part of the *robot_core* package. The *robot_core* package contains all the required nodes for the ATRV-mini to function and perform various tasks. Aided from the ROS community some ready-made packages were used.

The used nodes are described below.

3.4.1 /imu

This node is responsible for the communication with the IMU board and read all of its sensors' measurements. The node has been developed from scratch due to the lack of available ready packages for the Sparkfun Razor IMU M0 board that was used.

This node calculates and publishes robot's orientation (in quaternions form) angular velocity, and linear acceleration.

```
Node: /imu
Publishing topic: /imu
Message type:  sensor_msgs/Imu
```

3.4.2 GPS

An existing ROS package *nmea_navsat_driver* [34] for reading GPS values is used. This package's node reads standardized NMEA sentences as sent by the connected GPS and exports longitude, latitude, altitude, heading angle, linear velocity as well as other data.

```
Node: /nmea_serial_driver
Publishing topic: /fix
Message type: sensor_msgs/NavSatFix
Publishing topic: /vel
Message type: geometry_msgs/TwistStamped
Publishing topic: /time_reference
Message type: sensor_msgs/TimeReference
```

3.4.3 Camera

An existing ROS package *pixy_ros* for communicating with the Pixy camera was used [35]. This node publishes in pixel values x and y offset, width and height of the recognized object inside the frame of the camera.

```
Node: /pixy_node
Publishing topic: /block_data
Message type: pixy_msgs/PixyData
```

3.4.4 /analogread

This node has been developed to read the analog values provided to the BeagleBone Black. The analog values read are the battery voltage and consumption current, the solar cells voltage and charging current, and also the readings from the 4 ultrasonic sensors.

```
Node: /analogread
Publishing topic: /battery_state
Message type: sensor_msgs/BatteryState
Publishing topic: /solar_cell_state
Message type: sensor_msgs/BatteryState
Publishing topic: /ultrasonic_sensor_front_center
Message type: sensor_msgs/Range
Publishing topic: /ultrasonic_sensor_front_left
Message type: sensor_msgs/Range
Publishing topic: /ultrasonic_sensor_front_right
Message type: sensor_msgs/Range
Publishing topic: /ultrasonic_sensor_back_center
Message type: sensor_msgs/Range
```

3.4.5 /position

This node developed combines the data collected from the */imu* and */fix* topics of the IMU and GPS nodes respectively. After the combination of those data it publishes a *PoseStamped* type message which contains both spatial position and orientation.

```
Node: /position
Subscribed topic: /imu
Message type: sensor_msgs/Imu
Subscribed topic: /fix
Message type: sensor_msgs/NavSatFix

Publishing topic: /pose
Message type: geometry_msgs/PoseStamped
```

Position node also publishes in */tf* topic the transformations of the robot's position *base_link* (as described in chapter 3.3.2) relative to the *map* coordinate frame.

3.4.6 /wheel_controller

The *wheel_controller* node is in charge of controlling the wheels. It is subscribed to the */cmd_vel* topic where it receives messages in the form of linear and angular velocity vectors for the robot to execute. Based on the ATRV-mini's dimensions and its skid-steering kinematic model this node translates the received messages to commands sent via the serial port to the motor driver.

```
Node: /wheel_controller
Subscribed topic: /cmd_vel
Message type: geometry_msgs/Twist
```

3.4.7 /move_to_point

The *move_to_point* node is in charge of guiding the robot to move from its current position to the target position and orientation. It is subscribed to the */pose* topic in order to be aware of the robot's position and orientation at any given time and the */robot_state* topic in order to execute commands only when necessary. It is also subscribed to the */target_pose* topic where the target position and orientation are published for the robot to execute.

```
Node: /move_to_point
Subscribed topic: /pose
Message type: geometry_msgs/PoseStamped
Subscribed topic: /target_pose
Message type: geometry_msgs/PoseStamped
Subscribed topic: /robot_state
Message type: robot_core/RobotStates

Publishing topic: /cmd_vel
Message type: geometry_msgs/Twist
```

The pseudocode used in this node is shown below:

```
When new target pose is received
Do until target reached
   Calculate the distance between target and current pose
   If distance is greater than defined accuracy then
       Calculate angle of target pose relative to current pose
       If angle is less than defined accuracy then
           Publish move forward
       Else
           Publish turn left/right
   Else
       Target reached
```

3.4.8 /path_planner

This node is in charge of managing all the other functions developed in the other nodes on this package. It is subscribed to */path* topic where the desired path as an array of points is published. The robot moves through all the points of the path calling the previously mentioned */move_to_point* node consecutive times. This node is also subscribed to */battery_state*, */solar_cell_state* and */diagnostics* topics in order to have full view of the state of all major components of the robot.

```
Node: /path_planner
Subscribed topic: /battery_state
Message type: sensor_msgs/BatteryState
Subscribed topic: /solar_cell_state
Message type: sensor_msgs/BatteryState
Subscribed topic: /target_pose
Message type: geometry_msgs/PoseStamped
Subscribed topic: /robot_state
Message type: robot_core/RobotStates
Subscribed topic: /path
Message type:  nav_msgs/Path
Subscribed topic: /diagnostics
Message type: diagnostic_msgs/DiagnosticArray
```

```
Publishing topic: /target_pose
Message type: geometry_msgs/PoseStamped
Publishing topic: /robot_state
Message type: robot_core/RobotStates
```

3.4.9 /follow_object

When follow object state is enabled this node is executed. The node is subscribed in the */block_data* topic which receives data from the PixyCam. It contains the position of the defined color in the camera frame (in pixels) as well as the width and height of the object (in pixels).

```
Node: /move_to_point
Subscribed topic: /block_data
Message type: pixy_msgs/PixyData
Subscribed topic: /robot_state
Message type: robot_core/RobotStates
Publishing topic: /cmd_vel
Message type: geometry_msgs/Twist
```

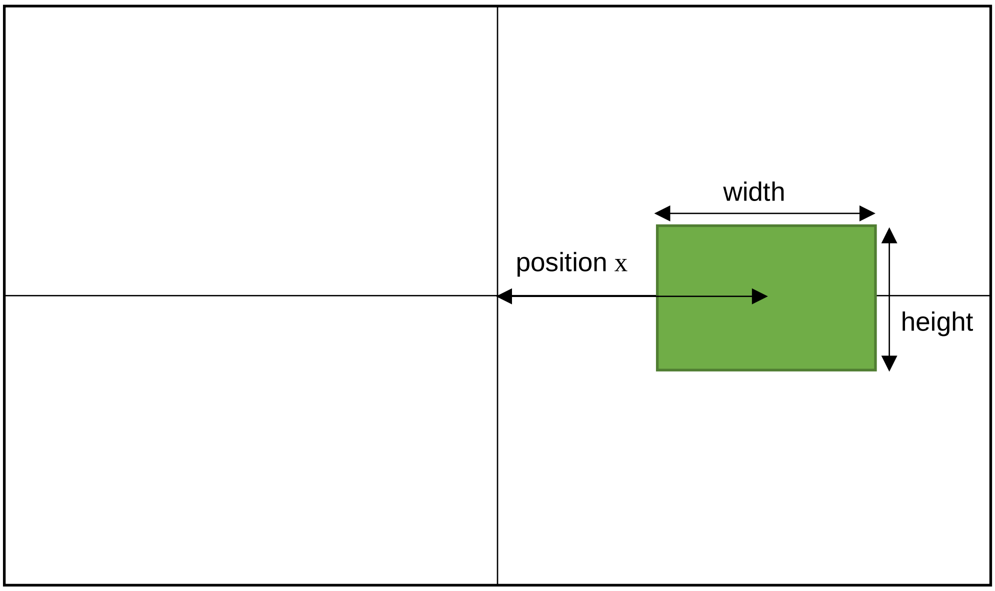


Figure 24 Pixy CMUCam Frame

The robot follows the recognized object using the pseudocode shown below:

```
When robot is set to the follow object state
Do until robot state changes
    Calculate position x of object
    If position x is greater than defined accuracy then
        Publish turn right
    Else
        Publish turn left
    If width of object is less than defined width then
        Publish move forward
    Else
        Publish move backward
```

The code developed for the *robot_core* package can be found on the bitbucket repository [36].

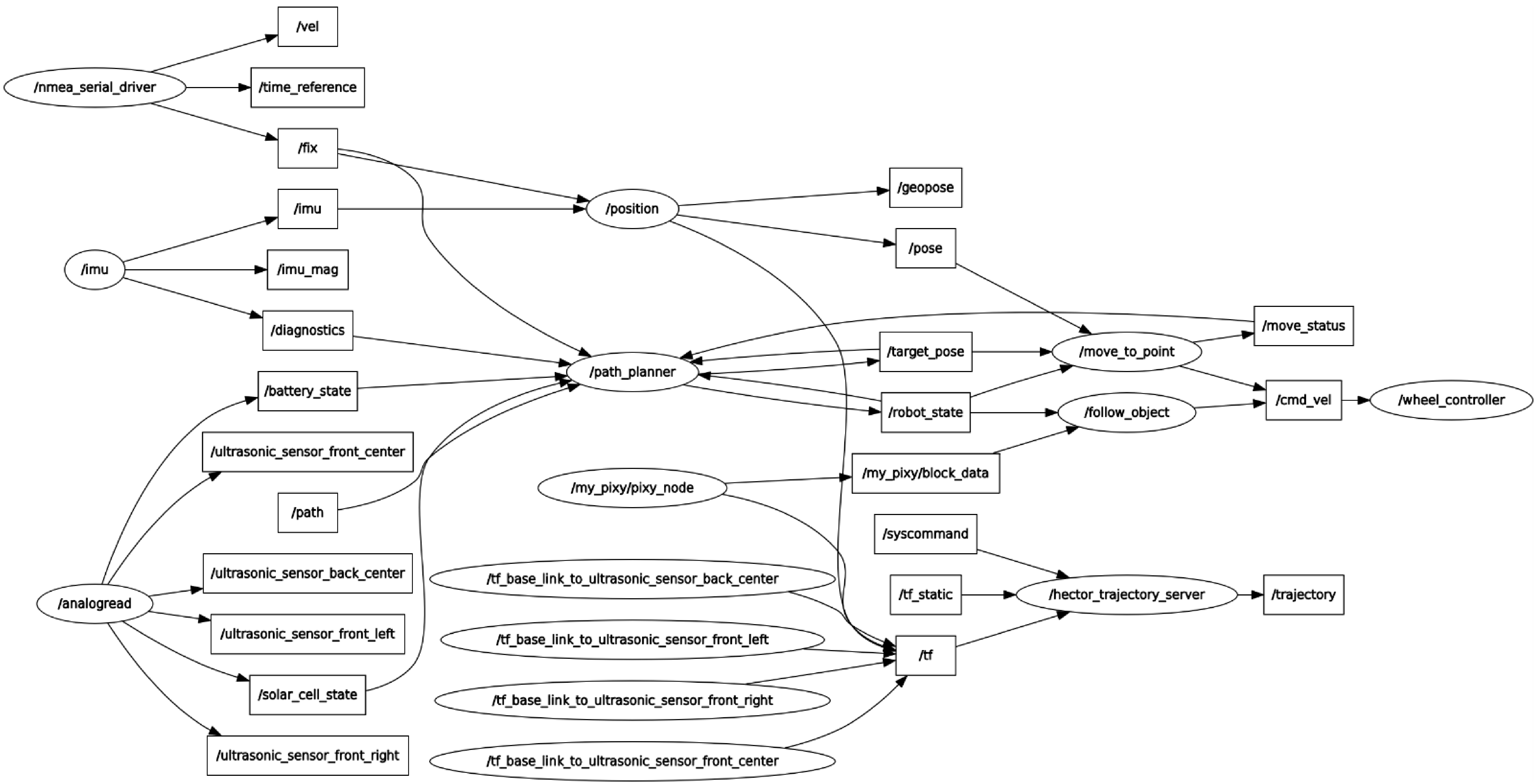


Figure 25 ROS Nodes and Topics

Chapter 4 Experiments

4.1 Remote Control

A Graphical User Interface (GUI) was developed so the user can remotely navigate the robot. The interface displays all important information such as commanded linear and angular speed, robot’s roll, pitch and yaw orientation and GPS readings.

4.2 Move to point

The first experiment made is to test the move to point functionality of the robot. On an allocated space two points are defined. The robot is placed on the first point’s location and then set to move from its current location to the defined point. As shown on the figure below with green color is the path for the robot to follow.

On the next figure the different followed paths are shown in dark pink color. The paths are visualization of the logged values from the GPS sensor during the experiment. As seen the deviation from the defined path is no greater than 1.8m as rated from the GPS manufacturer.

Figure 26 Move to Point with followed path

4.3 Waypoint Navigation

Waypoint navigation is also tested in an allocated open space. The coordinates of four points forming a rectangle are imported to the robot. Then the robot is called to execute a continuous loop on those points. The followed path is logged and visualized. As seen in (Figure 27) the deviation from the imported path is less than one meter.

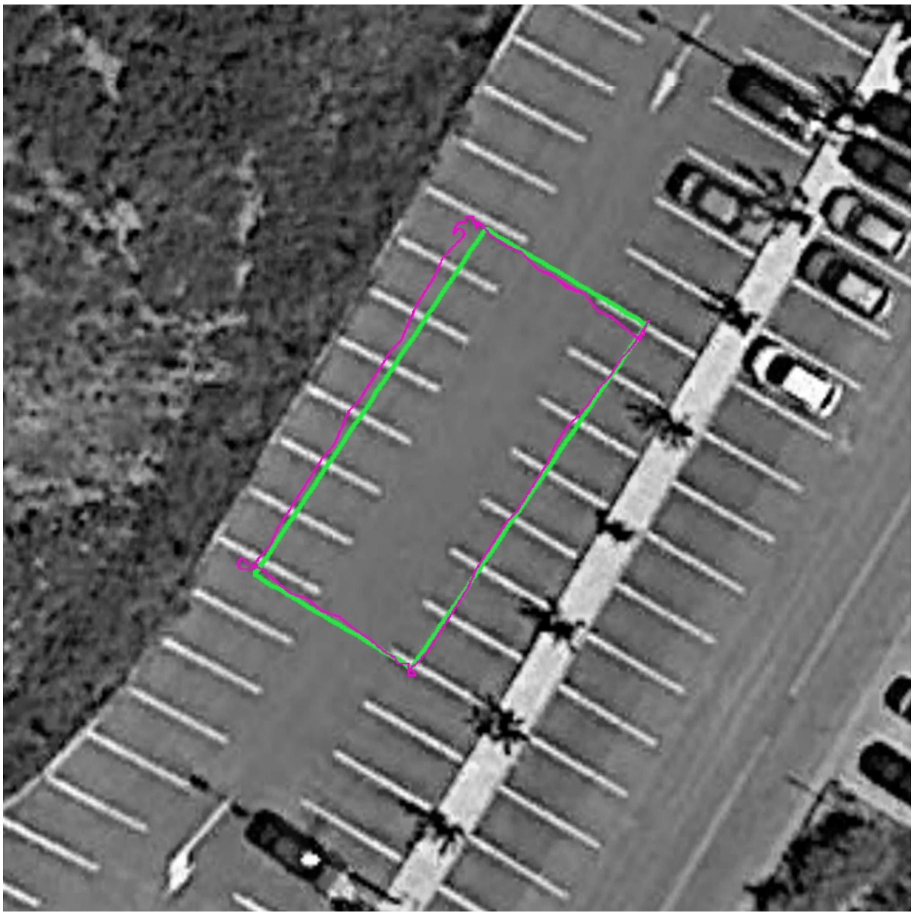


Figure 27 Follow path

4.4 Follow object

For the purposes of this experiment a vibrant green color is used to teach the Pixy CMUCam as set object. Then the robot was set to “follow object” mode. As seen in the two pictures bellow the robot is turning towards the object when the object is moving away from the center of its camera visual frame. The robot is also moving forwards or backwards when object dimensions sensed by the camera are decreasing or increasing respectively.

Figure 28 Turn and Follow Object

Video of the follow object experiment can be found on the following link [37]

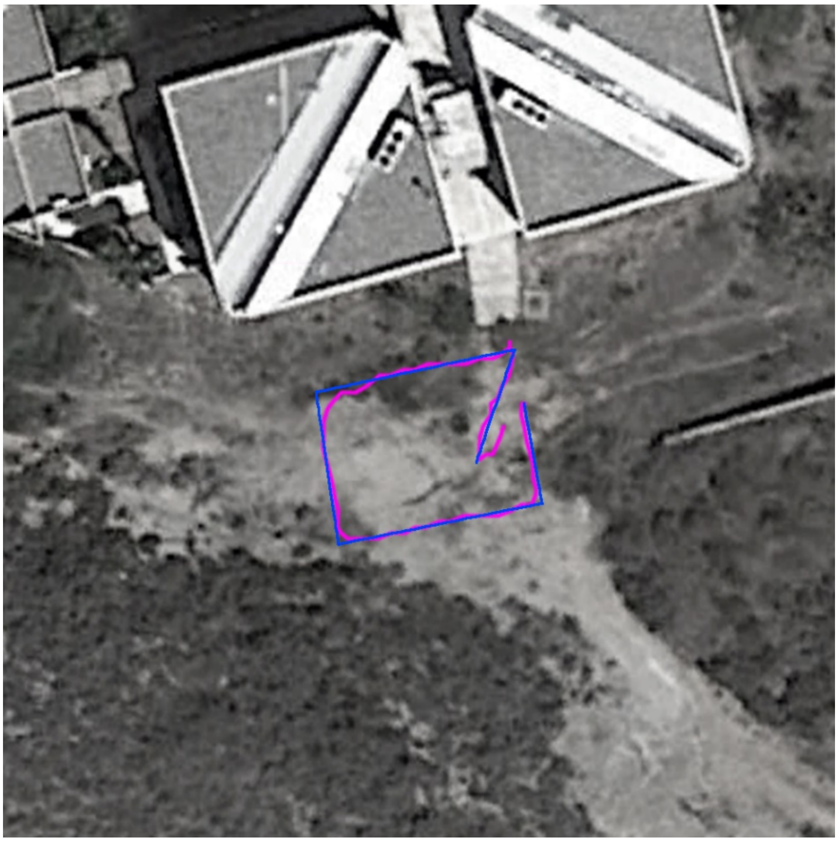


Figure 29 The paths followed by the target object and the robot are marked blue and pink respectively

4.5 Energy consumption

Robot's consumption in its original configuration was approximately 60W while idle. After the upgrade this consumption was reduced about 80%, leading to a total consumption of 12W, even though there is a certain boost in the overall processing power. Indicatively, the 29W consumption of the original onboard laptop was reduced to 1.2W of the BBB in the current configuration.

Table 2 Measured Consumption Current (A) on linear movement

Motor Speed	Spinning freely	On Pavement	On Gravel
Idle	0,13	0,13	0,13
10%	0,55	0,6	1,2
20%	0,85	1	1,4
30%	1,2	1,4	1,8
40%	1,65	1,9	2,8
50%	2,05	2,3	3,5
60%	2,5	2,8	3,8
70%	3	3,4	4,4
80%	3,5	3,8	5,2
90%	4	4,4	6,3
100%	4,4	4,8	6,8

The upgrade procedure resulted to a weight reduction of approximately 14%, even after the addition of the solar panels. The weight reduction leaded to a motor consumption decrease, although the propulsion system remained unchanged.

The enhancements above leaded to a notable operation time increase. This fact supported the decision of using reduced capacity batteries. Instead of the original 2x12V 12Ah, the lighter 2x12V 7Ah battery pack was used.

This new configuration was tested under the following scenario: every three minutes a 20 meter distance was covered repeatedly between two POIs, A and B. After three hours of testing, the remaining energy level was about 30%. This was a significant improvement compared to the original operating time of less than three hours.

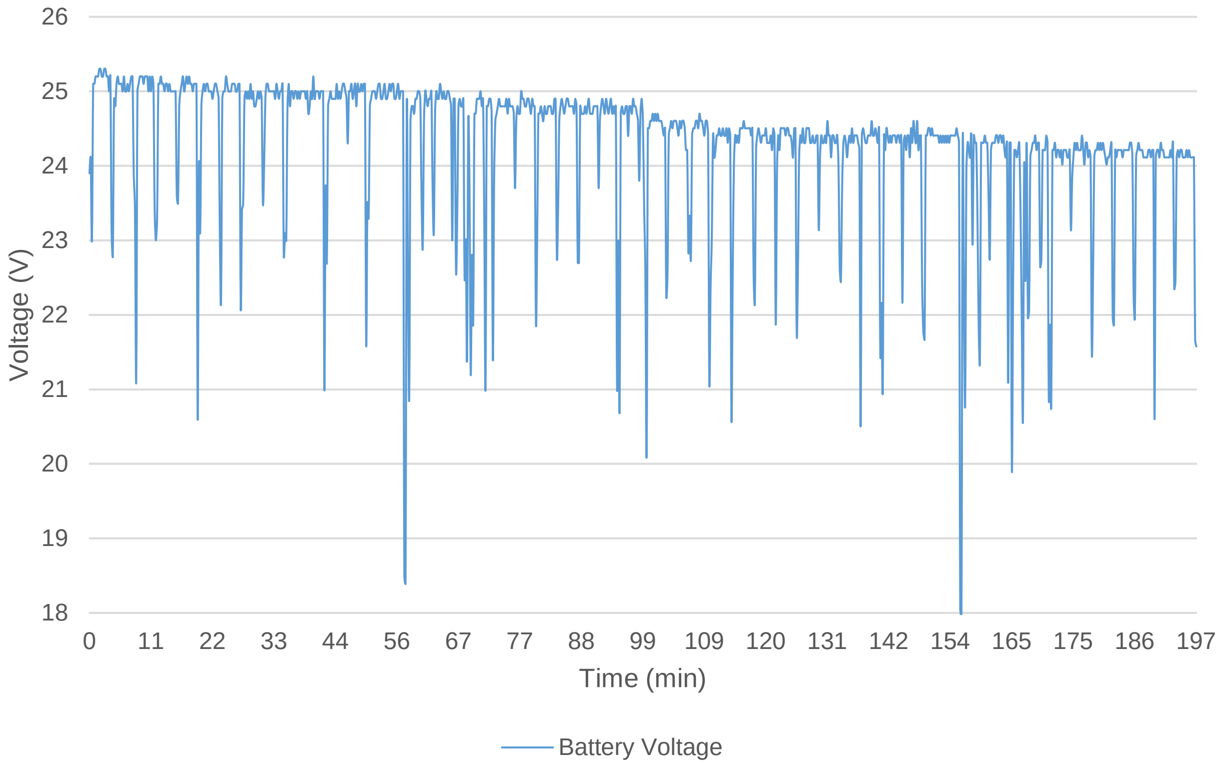


Figure 30 Battery voltage (V) during experiment time (min)

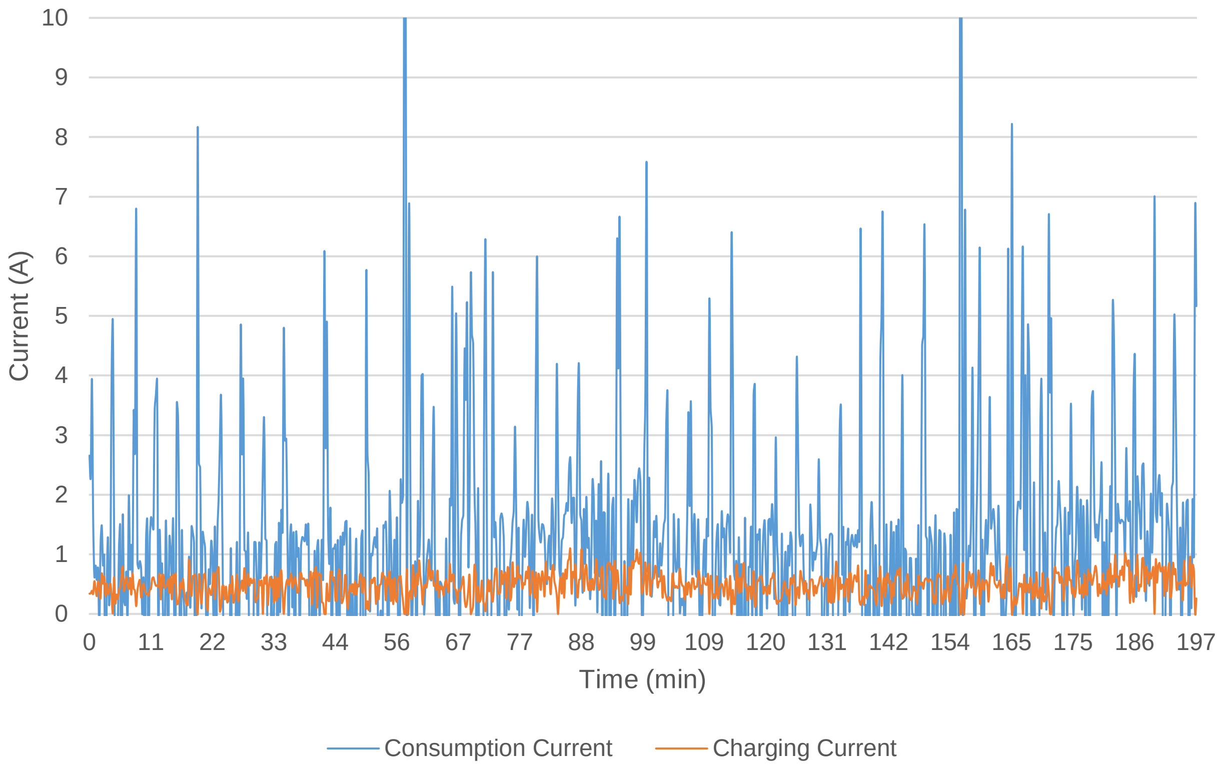


Figure 31 Consumption current and Charging current (A) during experiment time (min)

Chapter 5 Conclusion

ATRV-mini, a seventeen years old, retired outdoors robot, was reconstructed, with up to date robotic hardware and open source software solutions. It presented enhancements like increase of autonomy time and reduction of energy consumption and weight.

The components used in the upgrade procedure were of the shelf components and their cost was a fracture of the original equipment's cost. In contrast with early 2000s that there was limited access to commercially available components designed specifically for robotic use, leading the manufacturers to design and develop custom made components for their products.

In addition, the open source philosophy of the ROS framework used, reduced dramatically the development time of this project. Its provided tools aided in coding elegant solutions on solving various problems occurred during the development phase.

After the successful laboratorial and outdoors experimental testing, the robot proved to be adequate for undertaking basic robotic tasks. The robot is now at the disposal of the laboratory to be used for future research and educational purposes.

Given the above, it is obvious that the development of robotic applications nowadays is much more feasible for small teams with low budget than it was 17 years ago. It is no longer a privilege of high budget, multi personnel companies or laboratories.

Utilizing these tools a team can now focus its valuable resources in the research and development of real world robotic applications. And there are numerous of real world applications to focus on. Some examples are described below:

- Surveillance of non-trespassing areas
  The proposed autonomous robot can be used for surveillance of non-trespassing areas for human intrusion. Periodically checking Points of Interest such as violated doors or parts of fencing, or even distinguish human movement [5].
- Monitoring and maintenance of buildings and structures
  Replacing human monitoring of building or structures with the robot that will autonomously check for the selected POI. POI could be: fire extinguishers in their specified places, doors that should not be left open, state of valves, full garbage bins etc. [7]
- Agriculture monitoring
  The autonomous robots can be a remote assistant for agriculture applications. Defining whole trees or even crop plants as separate POI the robot will automatically recognize if the crop is ready to be collected and inform the user i.e. recognizing tomatoes when turning red notifying that are ready to collect.
- Fire fighting
  The robot can replace human surveillance for fire detection by using conventional or thermal camera [38].

References

[1] J. Vilca, A. Lounis and Y. Mezouar, "Optimal Multi-Criteria Waypoint Selection for Autonomous Vehicle Navigation in Structured Environment," *Intelligent Robotic Systems,* vol. 82, pp. 301-324, 2016.

[2] B. Sezen, "Modeling automated guided vehicle systems in material handling. Otomatiklestirilmi Rehberli Arac Sistemlerinin Transport Tekniginde Modellemesi," *Dou Universitesi Dergisi,* vol. 4, no. 2, pp. 207-216, 2011.

[3] Larissa Labakhua, Urbano Nunes, Rui Rodrigues and Fátima S. Leite, "Smooth Trajectory Planning for Fully Automated Passengers Vehicles: Spline and Clothoid Based Methods and Its Simulation," *Informatics in Control Automation and Robotics,* vol. 15, pp. 169-182, 2006.

[4] Maria M. Martins, Cristina P. Santos, Anselmo Frizera-Neto and Ramón Ceres, "Assistive mobility devices focusing on Smart Walkers: Classification and review," *Robotics and Autonomous Systems,* vol. 60, no. 4, pp. 548-562, 2012.

[5] Sascha A. Stoeter, Paul E. Rybski, Kristen N. Stubbs, Colin P. McMillen, Maria Gini, Dean F. Hougen and Nikolaos Papanikolopoulos, "A robot team for surveillance tasks: Design and architecture," *Robotics and Autonomous Systems,* vol. 40, no. 2-3, pp. 173-183, 2002.

[6] N. Metni and T. Hamel, "A UAV for bridge inspection: Visual servoing control law with orientation limits," *Automation in Construction,* vol. 17, pp. 3-10, 2007.

[7] J.-K. Oh, G. Jang, S. Oh, J. Ho Lee, B.-J. Yi, Y. Shik Moon, J. Seh Lee and Y. Choi, "Bridge inspection robot system with machine vision," *Automation in Construction,* vol. 18, pp. 929-941, 2009.

[8] H. Schempj, E. Mutschler, V. Goltsberg and W. Crowley, "GRISLEE: Gasmain Repair and Inspection System for Live Entry Environments," *The International Journal of Robotics Research,* vol. 22, no. 7-8, pp. 603-616, 2003.

[9] J. Lopez, D. Perez, E. Paz and A. Santana, "WatchBot: A building maintenance and surveillance system based on autonomous robots," *Robotics and Autonomous Systems,* vol. 61, pp. 1559-1571, 2013.

[10] "Intelligent Systems & Robotics Laboratory," 05 07 2017. [Online]. Available: http://www.robolab.tuc.gr/. [Accessed 05 07 2017].

[11] L. Doitsidis, K. P. Valavanis and N. C. Tsourveloudis, "Fuzzy Logic Based Autonomous Skid Steering Vehicle Navigation," in *CD-ROM Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington D.C., 2002.

[12] L. Doitsidis, K. P. Valavanis and N. C. Tsourveloudis, "Fuzzy logic based software control architecture for a skid steering vehicle," in *Proceedings of the Third International Workshop on Robot Motion and Control*, Bukowy Dworek, Poland, 2002.

[13] N. C. Tsourveloudis, L. Doitsidis and K. P. Valavanis, Autonomous Navigation of Unmanned Vehicles: A Fuzzy Logic Perspective, 2005, pp. pp:291-310.

[14] J. Aekaterinidis, K. Kostoulakis, L. Doitsidis, K. P. Valavanis and N. C. Tsourveloudis, "An Interface System for Real-Time Mobile Robot Environment Mapping using Sonar Sensors," *WSEAS Transactions on Systems,* Vols. vol. 4, no. 2, pp. 927-933, 2003.

[15] "Ubuntu," 15 07 2017. [Online]. Available: https://www.ubuntu.com/. [Accessed 15 07 2017].

[16] "Intel," 15 07 2017. [Online]. Available: https://www.intel.com. [Accessed 15 07 2017].

[17] "BeagleBone Black," 29 06 2017. [Online]. Available: https://beagleboard.org/black.

[18] "9DOF Razor IMU M0," 29 06 2017. [Online]. Available: https://www.sparkfun.com/products/14001.

[19] "NMEA 0183," 18 06 2017. [Online]. Available: https://en.wikipedia.org/wiki/NMEA_0183. [Accessed 18 06 2017].

[20] "Ultimate GPS Breakout," 29 06 2017. [Online]. Available: https://learn.adafruit.com/adafruit-ultimate-gps/overview.

[21] "LV-MaxSonar-EZ1," 29 06 2017. [Online]. Available: http://www.maxbotix.com/Ultrasonic_Sensors/MB1010.htm.

[22] "Pixy CMUcam5," 29 06 2017. [Online]. Available: http://www.cmucam.org/projects/cmucam5.

[23] "Sabertooth 2x12A," 29 06 2017. [Online]. Available: https://www.dimensionengineering.com/products/sabertooth2x12. [Accessed 29 06 2017].

[24] "Solar Cell 5.2W," 25 06 2017. [Online]. Available: https://www.sparkfun.com/products/retired/9241. [Accessed 25 06 2017].

[25] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System," *ICRA workshop on open source software,* vol. 3, no. 3.2, p. 5, 2009.

[26] "The Player Project," 2017. [Online]. Available: http://playerstage.sourceforge.net/. [Accessed 05 07 2017].

[27] "The Orocos Project," 2017. [Online]. Available: http://www.orocos.org/. [Accessed 05 07 2017].

[28] "Carmen Robot Navigation Toolkit," 2017. [Online]. Available: http://carmen.sourceforge.net/. [Accessed 05 07 2017].

[29] "Orca: Components for Robotics," 2017. [Online]. Available: http://orca-robotics.sourceforge.net/. [Accessed 05 07 2017].

[30] "The MOOS," 2017. [Online]. Available: http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php. [Accessed 05 07 2017].

[31] M. P. Tully Foote, ""REP 103: Standard Units of Measure and Coordinate Conventions"," 2010.

[32] W. Meeussen, ""REP 105: Coordinate Frames for Mobile Platforms"," 2010.

[33] "ECEF," 18 06 2017. [Online]. Available: https://en.wikipedia.org/wiki/ECEF. [Accessed 18 06 2017].

[34] ROS.org, "nmea_navsat_driver," 07 2017. [Online]. Available: http://wiki.ros.org/nmea_navsat_driver. [Accessed 07 2017].

[35] "pixy_ros," 07 2017. [Online]. Available: https://github.com/jeskesen/pixy_ros. [Accessed 07 2017].

[36] "reDevil," 05 07 2017. [Online]. Available: https://bitbucket.org/theonteg/redevil. [Accessed 05 07 2017].

[37] "ATRV-mini with Beaglebone Black, ROS and Pixy CMUcam follows object and visualizing in RViz," 29 04 2017. [Online]. Available: https://youtu.be/mKo34wQ2vJg?list=PLELCxDFJUhMffUqtIzQhzbbXAQwAe-Bm0. [Accessed 05 07 2017].

[38] N. I. Vitzilaios and N. C. Tsourveloudis, "UAS for Fire Management: State-of-the-art, Early Warning and Trends," in *MED 2009 Workshop on UAS Civilian Applications: Fire Detection, Forest Protection, Emergency Response*, Thessaloniki, Greece, 2009.

[39] Α. Τσαλατσάνης, "Οπτικό σύστημα του ATRV-MINI", 2001.

[40] Μ. Διαμαντάκης, "Ανάπτυξη αναδρομικού αλγόριθμου πλοήγησης με την βοήθεια συσκευών υπερήχου και χαρτογράφησης δωματίου με το έντροχο ρομπότ ATRV-mini", 2004.

[41] Α. Τσαλατσάνης, "Σύνθεση πληροφοριών αισθητήρων για την ασφαλή πλοήγηση έντροχου ρομποτικού οχήματος", 2004.

[42] Ε. Δοϊτσίδης, "Πλοήγηση και έλεγχος αυτόνομων επίγειων οχημάτων με πολλαπλούς αισθητήρες", 2002.

[43] E. Meristoudi, "Σχεδιασμός και προσδιορισμός λειτουργικών τμημάτων ρομποτικού οχήματος εξωτερικού χώρου", 2016.