SCIENTIFIC JOURNAL OF POLISH NAVAL ACADEMY
2020 (LXI) 4 (223)

sciendo

DOI: 10.2478/sjpna-2020-0015

MOBILE WHEELED ROBOT TO SUPPORT THE TASK OF THE ALARM SUB-UNIT

Daniel Powarzyński*

* *Polish Naval Academy, Faculty of Mechanical and Electrical Engineering, Smidowicza 69, Str., 81-127 Gdynia, Poland; email: danielpow@interia.pl;*

ABSTRACT

The article is a presentation, and detailed description of a mobile, vehicular robot whose task is to support the alarm sub-unit. The project was created in response to the increasing need for monitoring, and recognition of the areas. The robot's interface was created with the use of integrated development environments for Python. The software implementation was possible due to a minicomputer Raspberry Pi 4 B. The robot's frame is made out of components which are based on the main chassis. The robot is equipped with compatible sensors and cameras. Those, combined with the interface, are able to give a real-time preview of the area in which the robot is in.

This particular vehicular robot is designed to eliminate the risks caused by tasks of alarm sub-unit, by giving the real-time preview, and analysis of the currently watched area. In addition, it can be used to inspect soldiers in the containment zones, and to help with the identification of unknown objects.

Keywords:

mobile robot, raspberry pi, phyton, controlling a mobile robot

Research article

INTRODUCTION

Mobile robots are more common in civilian [3], and military case [14] scenarios. Because of their advantages, such as mobility, autonomy, manoeuvrability, and upgrading abilities they are a great option, if not an essential alternative, to human input. They can easily cover tasks traditionally made for people.

Advanced robots [17] can do more complicated tasks, and lower the possibility of unwanted loss at the same time. Thanks to high-quality materials [4], manipulators, sensors, and vision cameras those robots can be both fully functional, and precise. Their multitasking abilities are highly valued, especially when it comes to engineering, and minesweeping tasks. Artificial intelligence-based software [12], combined with cameras and sensors, allows the proper identification of objects and enemy firepower. It is also worth noting, that frequently mounted manipulators are essential in the context of explosive ordnance disposal missions. They are useful when it comes to neutralizing and moving explosive charges.

When it comes to mobile robots, PIAP PATROL [21] (Fig. 1) seems to be a perfect example. It is a medium-sized caterpillar track-type robot whose purpose is to detect any chemical, biological, radiological, and nuclear hazards. It can also identify and neutralize counter improvised explosive devices (C-IED). Due to the applied caterpillar drive, the robot can operate both inside, and outside, and achieves a top speed of 8km/h. The robot has 2 meters long, 6-graded manipulator, which can pick up, and lift objects to 22 kg. Also, the robot may have radiation sensors. They can spot, and identify chemical warfare agents, and fumes coming out of explosive charges.

Another example of a robot, that is out on the civil market, is SUMMIT-XL HL [22] (Fig. 2). It is a highly versatile mobile platform intended for indoor and outdoor use. Moving heavy loads is its designed use. The robot can be programmed and move autonomously, or manually with the aid of a remote control. Real-time preview is achievable through an installed PTZ camera. The robot is equipped with angle sensors too. They allow avoiding collisions by calculating the distance between the robot itself and an object.

The article presents programmed software, and engineering of vehicular mobile robots intended to support tasks of alarm sub-unit, as well as implementations of the system with real-time preview.

Fig. 1 PIAP PATROL [21]

Fig. 2 SUMMIT XL-HL [22]

ALARM SUB-UNIT

Alarm sub-unit is a military authority dedicated to:

- supervising, patrolling of the designated route (Fig. 3), and protecting military units
- securing crime scenes, and preventing the attempts to cover the traces of a crime, or to erase the evidence of crimes
- supporting and securing enterprises from the "Catalogue of extraordinary incidents, accidents, and breaches of military discipline reported to Operational Duty Service of the Secretary of Defence"
- securing intervention areas in cases of violation of military law, disturbance of safety, or supervising public order inside the military unit

All the above mentioned alarm sub-unit tasks can be supported by a mobile robot. Recording of any events allows soldiers to adapt better. At the same time, all records can be used as evidence, and provide materials to analyse any emergencies.

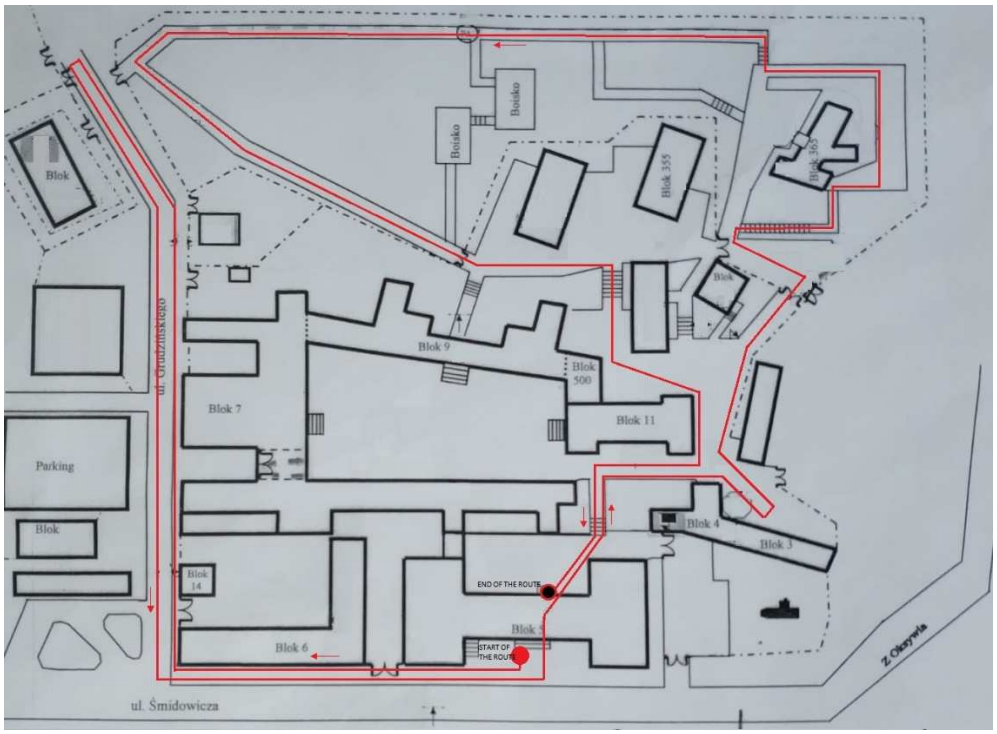

Fig. 3 External patrol route of a mobile robot

ROBOT STRUCTURE

The robot is based on a component frame. Because of that, every used hardware has to be lightweight, compatible with each other, and durable at the same time. Chassis is the main steel plate, and the most resilient item in the structure.

Front suspension with control system (Fig. 4 pt.1) has pieces that altogether allow connecting wheels, and shock-absorption system with the rest of the vehicle. The shock absorption in the suspension system is responsible for the stability and manoeuvrability, and is necessary when it comes to overcoming any obstacles. The suspension is built by connecting two co-working swing arms, shock absorbers springs, and stub axles. The structure of the rear suspension is similar to the one on the front, but because it is a rear-wheel drive, the whole suspension connects to the drive train (Fig. 4 pt.2). To make it move, putting wheel hubs, bearings, and semi-axis was required.

Drive train system is working through three cogwheels based on bearings. All of that sits inside a hermetic case made out of Polyamide. On the external ends of the shafts are flywheels connected to a driveshaft. 72-teeth disc wheel, mounted on

the main gear shaft, transfers the drive from the engine.

The main sub-components of the drive train are the electric brush motor RS-540SH (Fig. 4 pt. 3), and the drive wheel. They transmit the spin to a disc wheel. Combining optimised torque with well-balanced energy consumption allows obtaining both a good speed, and a long-lasting operative time. The engine is absorbing between 4.8-7.2 V with 23400rpm.

Mechanical speed controller (Fig. 4 pt.4) is responsible for accelerate, braking, and reversing. It keeps getting power from the battery, and sends it to the engine through the wires. The main parts of the controller are the spinning disk, reaction arm, and electric circuit. The mechanical speed controller connects to a resistor (Fig. 4 pt.5), which lowers the power voltage delivered from the battery to the engine. The resistor itself is surrounded by a radiator, which sends down the heat coming from electricity flow.

Servo-controllers are the ones responsible for fulfilling tasks, such as steering and turning wheels. The throttle servo-controller (Fig. 4 pt.7) converts electricity into a mechanical movement that is passing to the mechanical speed controller. The steering servo-controller (Fig. 4 pt.6) is responsible for front wheel movement. The power needed to run every component comes from the "Redox" nickelic-metal-hydrogen battery (Fig. 4 pt.8).

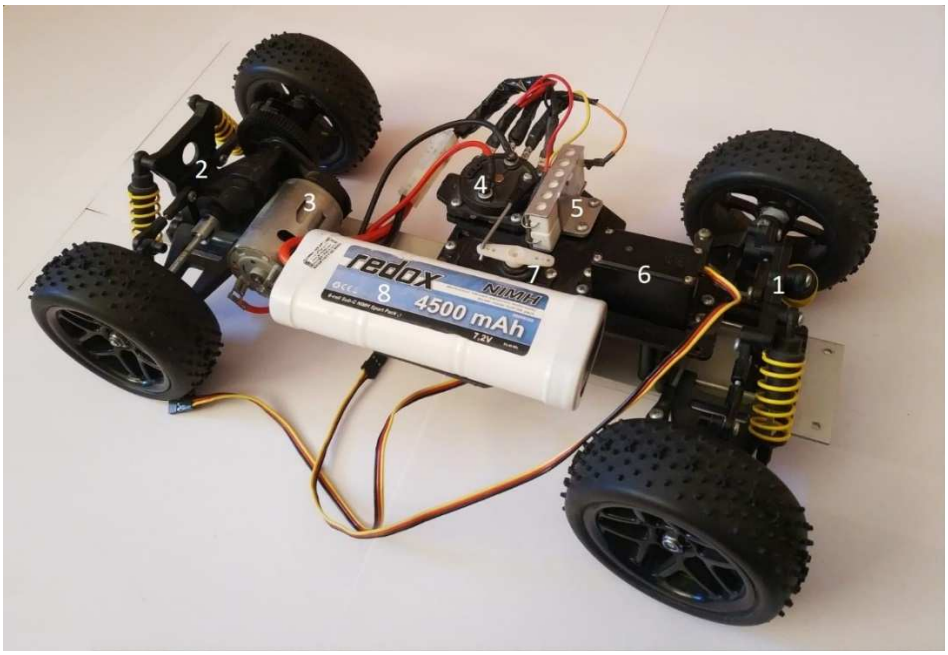

Fig. 4 Structural skeleton of a mobile robot

1 – front suspension system, **2** – rear suspension system with the drive train, **3** – electric engine RS-540SH, **4** – mechanical speed controller, **5** – resistor with heat sink **6** – steering servo controller, **7** – throttle servo controller, **8** – battery

ASSIST CONTROL SYSTEM – ULTRASONIC SENSORS SH-SR004

Two ultrasonic distance sensors HC-SR004 [20] are responsible for avoiding collision autonomously. Distance system measures the time between sending, and return of the ultrasonic wave. The microcontroller receives information about the distance. Aspects like sunlight, or dark colour of objects do not impact the way the sonar works. Sensors have 4 outlets VCC, GND, ECHO, and TRIG. To start measurement initiation, pin TRIG needs to receive 5V electrical pulse for 10 μs. Afterwards, the sensor sends eight pulses with 40 kHz frequency to measure the distance. A wave hits an encountered obstacle, and returns to the sensor. That signal is called echo. After the measurement, the pin ECHO receives the signal, and after that, the real measurement begins. How long it will take depends on the distance between the robot and obstacle. Range hovers between 150 μs to 25 ms. Receiving data takes up to 38 ms. The distance *d* is calculated according to the equation (1).

$$d = \frac{T * v}{2} \tag{1}$$

where:

d – distance [m]

T – the time interval between sending trigger signal, and receiving echo signal [s];

v – the velocity of sound [m/s].

The individual sensor can do the correct measure in between the 30-degree angle (Fig. 5). Most important is to set the sensors so that they do not overlap each other. If set incorrectly, they will distort the received signal. In consequence, the robot will do the wrong measurements, and it will not avoid an obstacle. Sensors should be placed opposite with +/- 45^{o} (Fig. 6) against the front suspension.

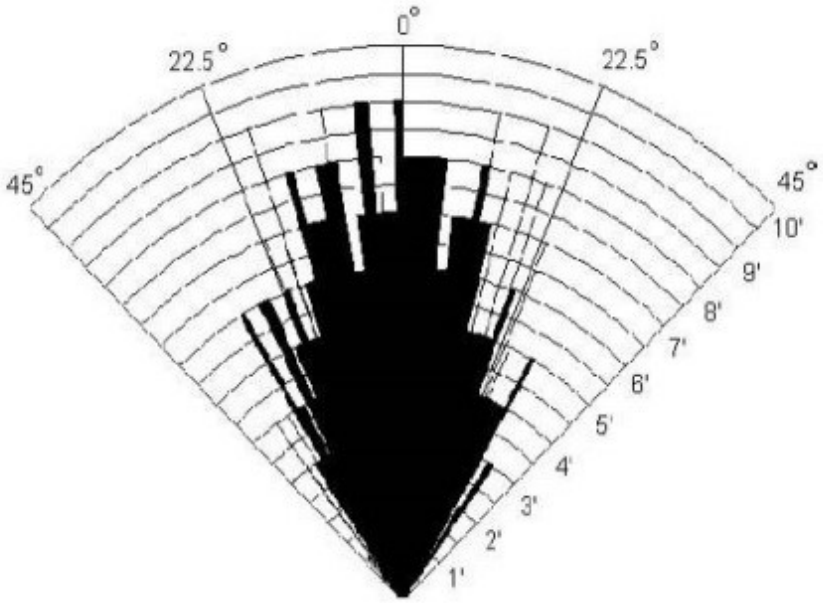

Fig. 5 Sensor measurement angle [20]

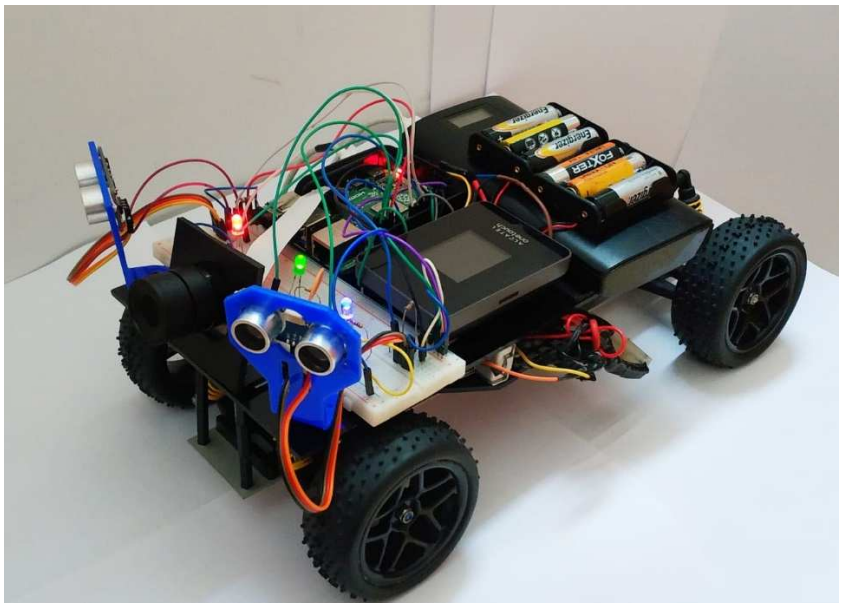

Fig. 6 Final construction of a mobile robot

IMPLEMENTATION

Hardware platform allows the user to control the robot by programming its movement. Responsible for that is a minicomputer Raspberry PI 4 B running on Broadcom BCM2711 network. It has 64 bit processor quad-core 64 ARM-8 CortexA72 with 1.5GHz frequency. The chosen programming language is Python, which allows the structure, function and object programming.

```
#Libraries used
import time
import sys
import RPi.GPIO as GPIO
import os
from echo import distance_thread
import threading
from distance import Czujnik
from pynput.keyboard import Listener, Key, Controller
#Pins designation
DISTANCE_LED_PIN = 40
READY_PIN = 38
CAM_ON_PIN = 36
SERVO_1_PIN = 10
SERVO_2_PIN = 11
#Pins initialization
GPIO.setmode(GPIO.BOARD)
GPIO.setup(DISTANCE_LED_PIN, GPIO.OUT)
GPIO.setup(READY_PIN, GPIO.OUT)
GPIO.setup(CAM_ON_PIN, GPIO.OUT)
GPIO.setup(SERVO_1_PIN,GPIO.OUT)
GPIO.setup(SERVO_2_PIN, GPIO.OUT)
#PWM setting on the pins of the servo-controller
servo1 = GPIO.PWM(SERVO_1_PIN,50)
servo2 = GPIO.PWM(SERVO_2_PIN, 54)
#Starting the servo-controller
servo1.start(0)
servo2.start(0)
czujnik = Czujnik() #Assing an object to communicate with the sensor
thread
#Assigning and starting the sensor handling thread
x = threading.Thread(target=distance_thread, kwargs={ "servo1" : servo1,
"servo2" : servo2, "czujnik" : czujnik })
x.start()
print("start procedure")
#Designation of the initial state of control buttons. The initial state
marked with "False" means the low state of control buttons
keyDownPress = False
keyUpPress = False
keyRightPress = False
keyLeftPress = False

#Led for the control system. Led signals start and readiness of  control
system
GPIO.output(READY_PIN, GPIO.HIGH)
keyboard = Controller()
```

Python is available on almost every operating system [16]. The one used in this project is the Raspberry Pi OS distributed by Linux. Time, Sys, Rpi GIPO, OS, Threading, and Pynput keyboard are all virtual libraries used in the project. Mater program executes things like robot steering, control of the remote control, and maintenance system.

```
#Button status reading function
def getCharPressed(key):
    try:
        return key.char
    except:
        return ""
#Global variable assignment
def on_press(key):
    global keyDownPress
    global keyUpPress
    global keyRightPress
    global keyLeftPress
    global czujnik
    print("Key pressed: {0}".format(key))
    if key == Key.esc: #The program will be turned off when "esc" button is used
        servo1.stop()
        servo2.stop()
        GPIO.output(READY_PIN, GPIO.LOW)
        czujnik.stop = True
   GPIO.cleanup()
        sys.exit()
    elif key == Key.down: #servo-controller motion initialization
        if not keyDownPress:
            servo1.ChangeDutyCycle(6.5)
            keyDownPress = True
    elif key == Key.up: #servo-controller motion initialization
        if not keyUpPress:
            servo1.ChangeDutyCycle(9)
            keyUpPress = True
    elif key == Key.left: #servo-controller motion initialization
        if not keyLeftPress:
            servo2.ChangeDutyCycle(5)
            keyLeftPress = True
    elif key == Key.right: #servo-controller motion initialization
        if not keyRightPress:
            servo2.ChangeDutyCycle(10.5)
            keyRightPress = True
    elif getCharPressed(key) == 's': #presing the "s" button activates the camera
        print("Starting the camera")
        os.system("/home/pi/Videos/RPi_Cam_Web_Interface/start.sh")
        GPIO.output(CAM_ON_PIN, GPIO.HIGH)
   elif getCharPressed(key) == 'd': #presing the "d" button turn off the camera
        print("Turn off the camera")
        os.system("/home/pi/Videos/RPi_Cam_Web_Interface/stop.sh")
        GPIO.output(CAM_ON_PIN, GPIO.LOW)
elif getCharPressed(key) == 'f': #presing the "f" button activates the assist control
system
        GPIO.output(DISTANCE_LED_PIN, GPIO.HIGH)
        czujnik.distanceThreadRun = True
    elif getCharPressed(key) == 'g': #presing the "g" button turn off the assist con-
trol system
        GPIO.output(DISTANCE_LED_PIN, GPIO.LOW)
        czujnik.distanceThreadRun = False
#Low status reading functions for control buttons
def on_release(key):
    global keyDownPress
    global keyUpPress
    global keyRightPress
```

```
    print("Key released: {0}".format(key))
    if key == Key.up:
        keyUpPress = False
    if key == Key.down:
        keyDownPress = False
    if key == Key.right:
        keyRightPress = False
    if key == Key.left:
        keyLeftPress = False

#Idle state of servo-controllers
    if key == Key.up or key == Key.down:
        servo1.ChangeDutyCycle(7.5)
    if key == Key.left or key == Key.right:
        servo2.ChangeDutyCycle(7.5)


try:
    with Listener(on_press=on_press, on_release=on_release) as li-
stener:
        listener.join()
except:
    print("ERROR")
    GPIO.cleanup()
```

A program, which executes assist control system, is based on a 'distance-thread' library. It is fully responsible for supporting both the sensors, and reactions of servo-controller to any given obstacle. If the distance between the robot, and obstacle is less than 25 cm, the program sends impulses to the steering control mechanism. That causes wheels to turn in the opposite direction, from where the signal is coming.

```
#libraries used
from distance import Czujnik
import RPi.GPIO as GPIO
import time

#pins designation
GPIO_TRIGGER = 12
GPIO_ECHO = 18
GPIO_TRIGGER2 = 16
GPIO_ECHO2 = 22

#Pins initialization
def init():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
    GPIO.setup(GPIO_ECHO, GPIO.IN)
    GPIO.setup(GPIO_TRIGGER2, GPIO.OUT)
    GPIO.setup(GPIO_ECHO2, GPIO.IN)
```

```
#Function for measuring the distance from sensors to obstacles
def distance(triggerPin, echoPin):
    GPIO.output(triggerPin, True)
    time.sleep(0.00001)
    GPIO.output(triggerPin, False)
    StartTime = time.time()
    StopTime = time.time()
    while GPIO.input(echoPin) == 0:
        StartTime = time.time()
    while GPIO.input(echoPin) == 1:
        StopTime = time.time()
    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2  #reference to the speed of li-
ght

return distance #return variable distance

#The function of handling sensors and the reaction of the steering servo-
controller to an obstacle
def distance_thread(servo1, servo2, czujnik):
    init()
    try:
        while not czujnik.stop: #The loop continues until interrupted by
the second script
            if not czujnik.distanceThreadRun:
                continue #In case of system shutdown, the program below
does not run
            time.sleep(0.01)
            dist = distance(GPIO_TRIGGER, GPIO_ECHO) #reading data from
sensor 1 (distance in cm)
            dist2 = distance(GPIO_TRIGGER2, GPIO_ECHO2) #reading data
from sensor 1 (distance in cm)
                print("obstacle 1")
                servo2.ChangeDutyCycle(10)
                time.sleep(1)
                servo2.ChangeDutyCycle(7.5)
            if dist2 < 25: #The condition will only be work if the di-
stance is less than 25 cm
                print("obstacle 2")
                servo2.ChangeDutyCycle(5)
                time.sleep(1)
                servo2.ChangeDutyCycle(7.5)
    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()
    print("Off system")
```

VISION SYSTEM

Vision system works through ArduCam OV5647 camera with sensor OV5647 (Fig. 6). The camera can zoom in, and set the focus, which is certainly an advantage when the robot is on the move. 5Mpx Image sensor supports 1080px/30fps. A camera module is equipped with lens LS-2718 CS with 4 mm focal length, 93degree

field of view, and 1.4 iris. That setup allows recording of the whole area in front of the robot.

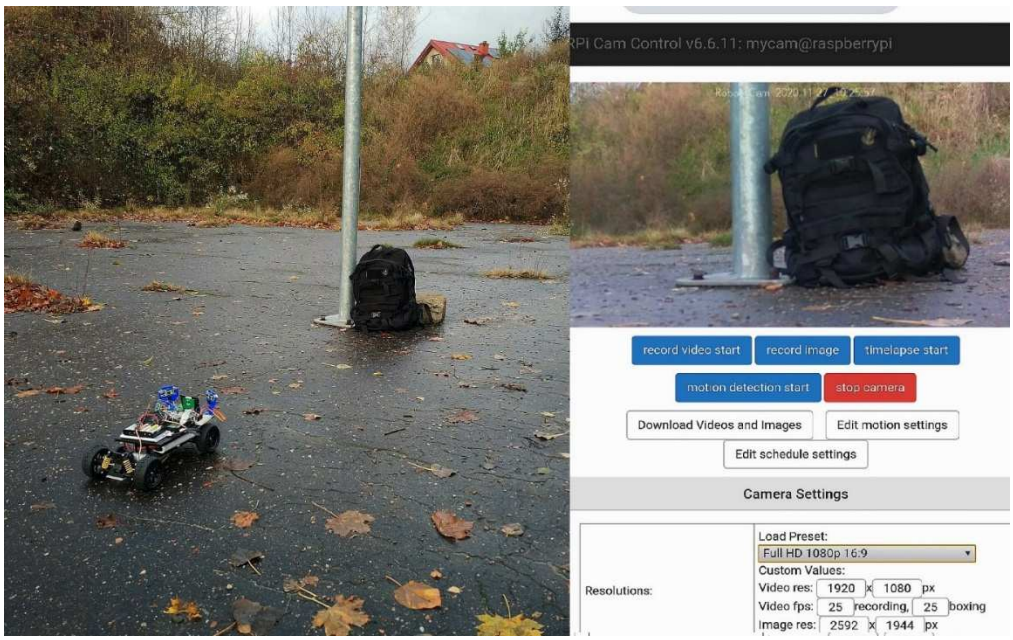

Fig. 7 Vision system - interface RPi Cam-Web

Realtime preview is possible through RPI-CAM-WEB (RPCWI) interface (Fig. 7). The expanded interface enables control of the settings of the camera through a browser script. RPCWI allows the user to record, and save all the data on the micro SD card, and then change every setting like saturation, or contrast. The camera interface is compatible with Linux. It shares a dedicated apache server that shows a preview of the camera view from all the mobile devices with any internet browser.

CONCLUSIONS

The presented vehicular mobile robot is good support for almost any of the alarm sub-unit. The mounted camera allows the user to spot, and identify any threats during the patrol. Because implementing universal software robot can be used for tasks inside the buildings where it can spot unidentified objects, or support the tasks of the soldiers inside the building. Future recourses will be focusing on enhancing the autonomous capabilities of the robot, as well as configuring the algorithms so multiple robots can operate more complex tasks

REFERENCES

[1] Eric Matthes, *Python Crash Course*, 2nd Edition: A Hands-On, Project-Based Introduction to Programming, Functions, 2019, pp.129-156

[2] Ignacy Dulęba. *Metody i algorytmy planowania ruchu robotów mobilnych i manipulacyjnych*

[3] Jung Hyun Choi, Kangwagye Samuel, Kanghyun Nam, Sehoon Oh, *An Autonomous Human Following Caddie Robot with High-Level Driving Functions*. Electronics 2020, *9*, 1516. https://doi.org/10.3390/electronics9091516

[4] Jurczyk Karolina, Piskur Paweł, Szymak Piotr, *Parameters Identification of the Flexible Fin Kinematics Model Using Vision and Genetic Algorithms*, Polish Maritime Research, *27*(2), 39-47, 2020, doi: https://doi.org/10.2478/pomr-2020-0025

[5] Lentin Joseph*, Nauka robotyki z językiem Python*, 2016

[6] Maciej Michałek, Dariusz Pazderski. *Sterowanie robotów mobilnych* : laboratorium, 2012

[7] Mariusz J. Giergiel, Zenon Hendzel, Wiesław Żylski. *Modelowanie i sterowanie mobilnych robotów kołowych,* PWN 2013

[8] Mark W. Spong, Seth Hutchinson, M. Vidyasagar, *Robot Modeling and Control*, 2nd Edition, 2020, pp. 365-373

[9] Matt Timmons-Brown, *Learn Robotics with Raspberry Pi: Build and Code Your Own Moving, Sensing, Thinking Robots*, 2019,

[10] Peter I. Corke, Robotics, *Vision and Control: Fundamental Algorithms in MATLAB*, 2nd ed. 2017 Edition

[11] Piotr Kulczycki, Józef Korbicz, Janusz Kacprzyk. *Automatyka, robotyka i przetwarzanie informacji*, PWN 2020

[12] Piotr Szymak, Pawel Piskur, Krzysztof Naus. (2020), *The Effectiveness of Using a Pretrained Deep Learning Neural Networks for Object Classification in Underwater Video, Remote Sensing*. 12.2020, doi: 10.3390/rs12183020.

[13] Piskur Pawel; Szymak Piotr; Jaskólski Krzysztof; Flis, Leszek; Gąsiorowski Marek, *Hydroacoustic System in a Biomimetic Underwater Vehicle to Avoid Collision with Vessels with Low-Speed Propellers in a Controlled Environment. Sensors*, 2020,

https://doi.org/10.3390/s20040968

[14] Przybylski Michał, Szymak Piotr, Kitowski Zygmunt, Piskur Pawel, *Comparison of Different Course Controllers of Biomimetic Underwater Vehicle with Two Tail Fins*. In: Bartoszewicz A., Kabziński J., Kacprzyk J. (eds) Advanced, Contemporary Control. Advances in Intelligent Systems and Computing, vol 1196. Springer, Cham., 2020,

https://doi.org/10.1007/978-3-030-50936-1_125

[15] Spyros G. Tzafestas, *Introduction to Mobile Robot Control, Mobile Robot Control V: Vision-Based Method,* 2014, pp. 319-351.

[16] Steven F. Lott *Python : programowanie funkcyjne*, 2019

[17] Wei Bin, *A Low Cost Introductory Platform for Advanced Robotic Control*. In: Venture G., Solis J., Takeda Y., Konno A. (eds) ROMANSY 23 - Robot Design, Dynamics and Control. ROMANSY 2020. CISM International Centre for Mechanical Sciences (Courses and Lectures), vol 601. Springer, Cham., 2020, https://doi.org/10.1007/978-3-030-58380-4_18

[18] Wiktor Hudy, Kazimierz Jaracz, *Laboratorium automatyki i robotyki*, 2013

[19] Wojciech Kaczmarek, Jarosław Panasiuk, Szymon Borys. *Środowiska programowania robotów*, PWN 2020

[20] www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf

[21] www.piap.pl/produkt/roboty-mobilne-do-zastosowan-specjalnych/

[22] www.robotnik.eu/products/mobile-robots/summit-xl-hl/

MOBILNY ROBOT KOŁOWY DO WSPARCIA PODODDZIAŁU ALARMOWEGO

STRESZCZENIE

W artykule zaprezentowano mobilnego robota kołowego do wsparcia pododdziału alarmowego. Projekt został zrealizowany ze względu na rosnące potrzeby identyfikacji, kontroli oraz obserwacji. Cześć programistyczna została zaimplementowana w oparciu o minikomputer Raspberry Pi 4 B, z wykorzystaniem środowiska programistycznego PYTHON oraz dostępnych bibliotek. Cześć konstrukcyjna bazuje na szkielecie komponentowym zbudowanym na głównym elemencie podwoziowym. W projekcie wykorzystano kompatybilne czujniki uzupełniające funkcjonalność robota oraz kamerę z obiektywem, która wraz z zastosowanym interfejsem realizuje pogląd obrazu w czasie rzeczywistym.

Robot przeznaczony jest do eliminowania zagrożeń wynikających z zadań pododdziału alarmowego poprzez wizyjną analizę pola obserwacji, dodatkowo umożliwia nadzór żołnierzy przebywających w strefach izolacyjnych oraz identyfikacje nierozpoznanych obiektów.

Słowa kluczowe:

robot mobilny, raspberry pi, phyton, sterowanie robotem mobilnym