UDC 004.89

DESIGN OF MINE-DETECTING ROBOT USING YOLOv8 OBJECT DETECTION MODEL

A.Karpenko
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, pr. Nauki, 14
E-mail: andrii.karpenko2@nure.ua

Abstract: Methods of developing a demining robot, based on Android application control and YOLOv8 image detection algorithm, are uncovered in this article.

Keywords: computer vision, demining, explosives, landmine, mine, object detection, robot, robotics, Arduino, MIT App Inventor, YOLOv8.

РОЗРОБКА МІННОПОШУКОВОГО РОБОТА З ВИКОРИСТАННЯМ СИСТЕМИ РОЗПІЗНАВАННЯ ОБРАЗІВ МОДЕЛІ YOLOv8

А.С.Карпенко
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: andrii.karpenko2@nure.ua

Анотація: В даній статті описано спосіб побудови міннопошукового робота з системою комп’ютерного зору на базі моделі нейронної мережі YOLOv8 та системою керування від Android-додатку.

Ключові слова: комп’ютерний зір, розмінування, вибухівка, вибухонебезпечний предмет, міна, розпізнавання об'єктів, робот, робототехніка, Arduino, MIT App Inventor, YOLOv8.

INTRODUCTION. Previously [1], the significance of modern demining robot development, as well as demining solutions were mentioned. In short, due to violations of human rights, nowadays it is common to find landmines or other explosives on certain areas. This issue is especially important in Ukraine: country, that became the most mine-flooded in world due to russian military intervention. As a result, it is of great importance to clean up the landscape from explosives.

From the perspective of safety, robots are well suited for demining tasks: better to lose a machine rather than human life. However, robotic solutions have their own disadvantages, such as limited mobility and huge sizes. In this way, it is still possible to develop a better alternative to existing robotic solutions.

HARDWARE SELECTION. The main goal of robot is both move on a flat terrain and be capable to detect the explosives, and so the hardware selection was the main priority during development of new robot. In this way, the structure of robot was planned and developed according to figure 1.

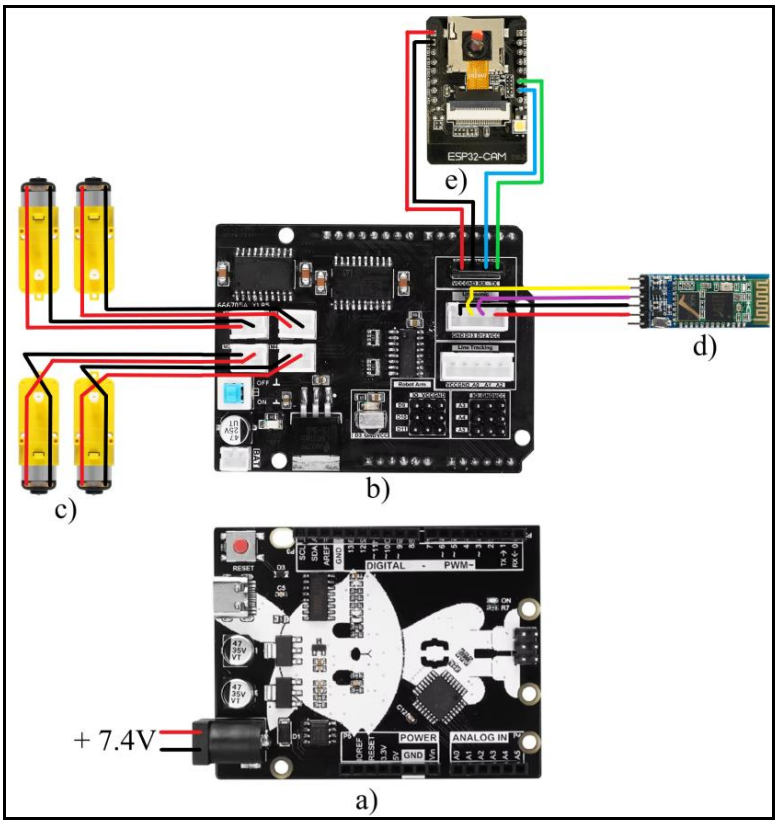
The project does not involve sophisticated kinematics, like legged movement, precise allocation and other, so instead of building the carcasses and choosing the electronic parts from scratch, existing solution in a form of one of TSCINBUNY robots [2].

The schematic includes Arduino-like controller (figure 1.a), that can accept the 7.4V charge and reallocate it to servos (figure 1.c) via a specific shield (figure 1.b). The movement controls are transported via a Bluetooth module (figure 1.d), while the image from ESP-32CAM camera (figure 1.e) is sent to the laptop via webpage. Even though camera is physically connected to the controller, the only type of their communication is a set of messages for launching the camera during the launch of robot.

SOFTWARE DEVELOPMENT. In this way, there are two subsystems to be programmed: movement and object detection ones.

To move the developed machine, an operator should have sent such a message to the Bluetooth module, that would correspond to one of the movement instructions. Furthermore, it is not enough to send only one message. This process is represented on figure 2: there, blue color represents the events in physical world, red one represents instructions, acquired by robot (M for movement, S for stop, Noth for nothing, when connection is lost), black cyclogram represents commands, executed by robot.

It similar projects [3,4,5], usually the movement of robot changes during the touching button up and down (figure 2.a). It is convenient for general projects because of simplicity in code, however it does not suit military-related applications due to lack of noise protection [6,7].



a) – TSCINBUNY controller;
b) – McNamham wheel manipulator trolley shield;
c) – gear motors;
d) – HC-08 Bluetooth sensor;
e) ESP32-CAM.

Figure 1 – Connection schematics of the developed robot

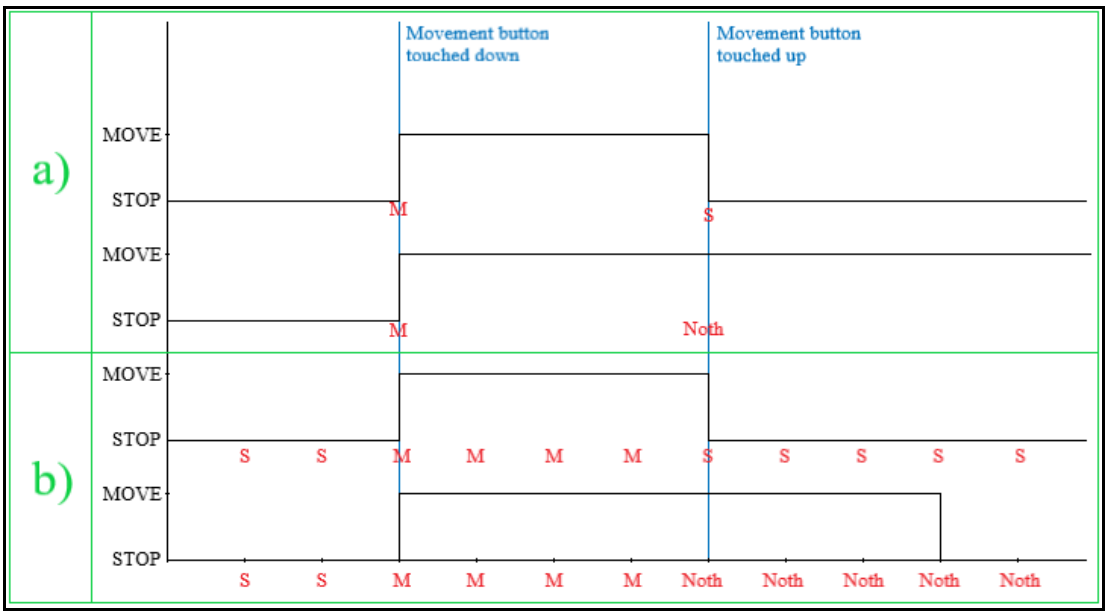
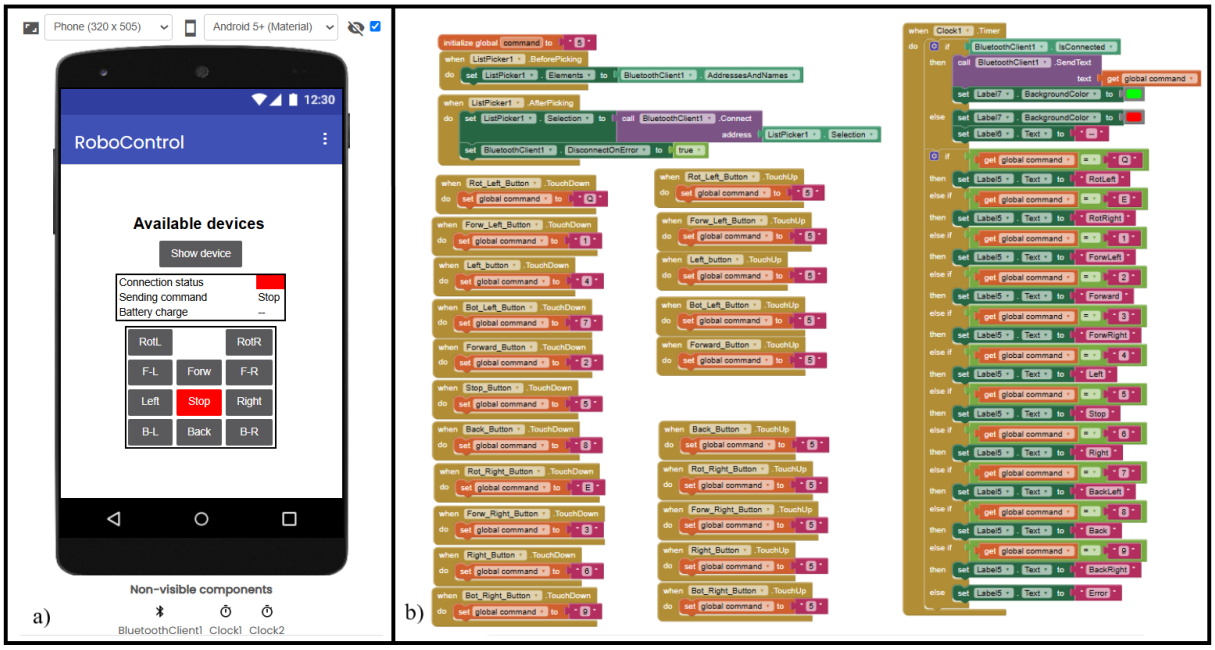


Figure 2 – Cyclograms for instruction execution for different robotic movement algorithms

If robot does not get the touch-up instruction (“stop”), then it will keep moving. Moreover, such behavior is unpreventable due to impossibility to distinguish the “Noth” instruction from user inactivity.

Thus, it is more reliable to perform instructions according to figure 2.b, where certain instruction is constantly sent on controller once in a particular time interim. Here, if no instructions get to the controller, then timer can be launched to make robot stop after certain time.

According to algorithm mentioned above, an Android application was developed to send instructions to controller (figure 3).



a) – layout; b) – code

Figure 3 – Android application for movement control

After getting the message via a virtual serial port, robot must process acquired instructions, which is done according to method, mentioned in [3]. Also, the movement stop after connection lost was considered. As a result, the code from figure 4 was uploaded into controller.

```
#include<SoftwareSerial.h>
#include"SafeString.h"
// PWM control pin
#define PWM1_PIN            5
#define PWM2_PIN            6
// 74HCT595N Chip pins
#define SHCP_PIN            2                       // The displacement of the clock
#define EN_PIN              7                       // Can make control
#define DATA_PIN            8                       // Serial data
#define STCP_PIN            4                       // Memory register clock

const int LeftUpBack       = 128;                     // forward
const int LeftUpForw       = 64;                      // back
const int LeftDownForw     = 32;                      // left translation
const int LeftDownBack     = 16;                      // Right translation
const int RightDownBack    = 8;                       // Upper left mobile
const int RightDownForw    = 1;                       // Lower left mobile
const int RightUpForw      = 4;                       // Upper right mobile
const int RightUpBack      = 2;                       // The lower right move


const int Forward       = LeftUpForw | LeftDownForw | RightDownForw |RightUpForw;      // forward
const int Backward      = LeftUpBack | LeftDownBack | RightDownBack | RightUpBack;     // back
const int Turn_Left     = LeftUpBack | LeftDownForw | RightUpForw | RightDownBack;                          // left translation
const int Turn_Right    = LeftUpForw | LeftDownBack | RightUpBack | RightDownForw;                         // Right translation
const int Top_Left      = LeftDownForw | RightUpForw;                              // Upper left mobile
const int Bottom_Left   = LeftUpBack | RightDownBack;                             // Lower left mobile
const int Top_Right     = LeftUpForw | RightDownForw;                              // Upper right mobile
const int Bottom_Right  = LeftDownBack | RightUpBack;                              // The lower right move
const int Stop          = 0;                                    // stop
const int Contrarotate  = LeftUpBack | LeftDownBack | RightDownForw | RightUpForw;                          // Counterclockwise rotation
const int Clockwise     = LeftUpForw | LeftDownForw | RightDownBack | RightUpBack;                           // Rotate clockwise
```

Figure 4 – Controller code for movement of robot, page 1

```
SoftwareSerial bluet_com(12, 13); // RX and TX on board, TX and RX on module
//SoftwareSerial battery_com(9,10); // RX and TX on board, TX and RX on module
char contr_letter = ' ';
createSafeString(batterymes,10);
unsigned long currentmillis;
unsigned long previousmillis;

void setup()
{
    Serial.begin(9600);
    bluet_com.begin(9600);
    pinMode(SHCP_PIN, OUTPUT);
    pinMode(EN_PIN, OUTPUT);
    pinMode(DATA_PIN, OUTPUT);
    pinMode(STCP_PIN, OUTPUT);
    pinMode(PWM1_PIN, OUTPUT);
    pinMode(PWM2_PIN, OUTPUT);
}

void loop()
{
    currentmillis = millis();
    if(bluet_com.available())
    {
        Serial.println("got in");
        contr_letter = bluet_com.read();
        Serial.println(contr_letter);
        previousmillis = currentmillis;

        if(contr_letter=='2') {
          /* Forward */
          Motor(Forward, 250);
        }
        else if(contr_letter == '8'){
          /* Backward */
          Motor(Backward, 250);
        }
        else if(contr_letter == '4'){
          /* Turn_Left */
          Motor(Turn_Left, 250);
        }
        else if(contr_letter == '6'){
          /* Turn_Right */
          Motor(Turn_Right, 250);
        }
        else if(contr_letter == '1'){
          /* Top_Left */
          Motor(Top_Left, 250);
        }
        else if(contr_letter == '3'){
          /* Top_Right */
          Motor(Top_Right, 250);
        }
        else if(contr_letter == '7'){
          /* Bottom_Left */
          Motor(Bottom_Left , 250);
        }
        else if(contr_letter == '9'){
          /* Bottom_Right */
          Motor(Bottom_Right, 250);
        }
        else if(contr_letter == 'E'){
          /* Clockwise */
          Motor(Clockwise, 250);
        }
        else if(contr_letter == 'Q'){
          /* Contrarotate */
          Motor(Contrarotate, 250);
        }
        else if(contr_letter == '5'){
          /* Stop */
          Motor(Stop, 250);
        }
    }


    if ((currentmillis-previousmillis)>1000)
    {
      Motor(Stop, 250);
    }

    delay(100);
}

void Motor(int Dir, int Speed)
{
    digitalWrite(EN_PIN, LOW);
    analogWrite(PWM1_PIN, Speed);
    analogWrite(PWM2_PIN, Speed);

    digitalWrite(STCP_PIN, LOW);
    shiftOut(DATA_PIN, SHCP_PIN, MSBFIRST, Dir);
    digitalWrite(STCP_PIN, HIGH);
}
```

Figure 4 – Controller code for movement of robot, page 2

At the same time, robot must also transmit the video to computer for object detection tasks. To do it, a code from [8] was applied to transfer the image from ESP32-CAM to the local webpage. Then, the code was processed in PyCharm, where the image is processed with a pretrained YOLOv8 model.

To perform an object detection, a neural net must be trained and infused in a system [9,10]. This involves preparing training, cross validation and test data, followed by training prosses itself and ending up with passing the image through image detection application. The dataset of explosives was taken from [11]. Then, model was trained in Google Collab according to [12]. Lastly, the video from web page was extracted according to [3], processed and displayed (figure 5 and 6).

```python
import cv2
import urllib.request
import numpy as np
from ultralytics import YOLO
import supervision as sv

# Replace the URL with the IP camera's stream URL
url = 'http://192.168.0.121/cam-hi.jpg'
cv2.namedWindow( winname: "live Cam Testing", cv2.WINDOW_AUTOSIZE)

 # Create a VideoCapture object
cap = cv2.VideoCapture(url)


# Check if the IP camera stream is opened successfully
if not cap.isOpened():
    print("Failed to open the IP camera stream")
    exit()

model = YOLO("C:/ENGENEERING/4cour_dipl/Neural_Net_code/Detector/last.pt")
box_annotator = sv.BoundingBoxAnnotator(
    thickness=2
)

threshold = 0.3

# Read and display video frames
while True:
    # Read a frame from the video stream
    img_resp = urllib.request.urlopen(url)
    imgnp = np.array(bytearray(img_resp.read()), dtype=np.uint8)
    im = cv2.imdecode(imgnp, -1)

    results = model(im)[0]
    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result

        if score > threshold:
            cv2.rectangle(im, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2)
            cv2.putText(im, text: f'{results.names[int(class_id)]}, {score:.2f}', org: (int(x1), int(y1 - 10)),
                        cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, color: (0, 255, 0), thickness: 2)

    cv2.imshow( winname: 'live Cam Testing', im)
    key = cv2.waitKey(5)
    if key == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Figure 5 – Python code for processing and displaying image

Figure 6 – Results of detection

CONCLUSION. Even though neural network successfully detects most of explosives, there are a lot of false negatives in the detections. Also, the developed robot lacks battery monitoring system and chassis for dust protection, so there are still ways of developing the project.

REFERENCES

1. Karpenko A. Overview at modern mine detecting robots. *Aded-2024.* 2024. No. 2. P. 181–185. URL: https://drive.google.com/file/d/1u_rEiXZx-TIR7n1sBHkL_OuFF5w7kOb1/view (date of access: 06.04.2025).

2. ESP32 camera robot for arduino uno starter kit. *TSCINBUNY*. URL: https://tscinbuny.com/products/tscinbuny-esp32-robot-for-arduino-uno-starter-kit-programmable-robot-educational-kit-4wd-60mm-omni-directional-wheel-chassis-with-wifi-app-obstacle-avoidance-line-tracking-smart-car-set (date of access: 06.04.2025).

3. Viral Science - The home of Creativity. Arduino mecanum wheel robot car | all direction smartphone controlled, 2021. *YouTube*. URL: https://www.youtube.com/watch?v=SYtFCyF44h0 (date of access: 06.04.2025).

4. Srishti Robotics. Bluetooth controlled robot using arduino uno and MIT APP inventor, 2021. *YouTube.* URL: https://www.youtube.com/watch?v=QC6TDIduhfg (date of access: 06.04.2025).

5. Muhammad Ansar. Arduino Robot Car Control using HC-05 Bluetooth | mit app inventor for android apk, 2019. *YouTube*. URL: https://www.youtube.com/watch?v=HBQIH98wmHs (date of access: 06.04.2025).

6. Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В., Новоселов С. П., Демська Н. П. Проектування мобільних маніпуляційних роботів: Монографія. – Х. :, 2022. – 427 с.

7. Nevliudov I., Tsymbal O., Bronnikov A. Fuzzy decision-making for intelligent robotic system. *Optoelectronic devices in robotic systems*. Cham, 2022. P. 227–255. URL: https://doi.org/10.1007/978-3-031-09791-1_9 (date of access: 07.04.2025).

8. ESP32 CAM object detection & identification with opencv. *How To Electronics*. URL: https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/ (date of access: 07.04.2025).

9. Queries classification using machine learning for implementation in intelligent manufacturing / V. Bortnikova et al. *Methods and tools in cad selected issues* / ed. by B. Butryło. Białystok, 2021. P. 63–74.

10. Yevsieiev V., Maksymova S., Alkhalaileh A. Mobilenetv2 neural network model for human recognition and identification in the working area of a collaborative robot. *Multidisciplinary journal of science and technology*. 2024. Vol. 4, no. 8. P. 5–12. URL: https://www.mjstjournal.com/index.php/mjst/article/view/1783 (date of access: 07.04.2025).

11. Demining object detection dataset by trashsorting. *Roboflow*. URL: https://universe.roboflow.com/trashsorting-gg1jy/demining-rpd9q (date of access: 07.04.2025).

12. Computer vision engineer. Train Yolov8 object detection on a custom dataset | Step by step guide | Computer vision tutorial, 2023. *YouTube*. URL: https://www.youtube.com/watch?v=m9fH9OWn8YM (date of access: 07.04.2025).

13. Chala, O., Yevsieiev, V., Maksymova, S., & Abu-Jassar, A. (2025). MATHEMATICAL MODEL BASED ON MULTI-AGENT REINFORCEMENT LEARNING (MARL) AND PARTIALLY OBSERVABLE MARKOV DECISION PROCESS (POMDP) FOR MODELING CARGO MOVEMENT FOR A MOBILE ROBOTS GROUP. *Multidisciplinary Journal of Science and Technology*, *5*(4), 480-489.

14. Maksymova, S., Yevsieiev, V., & Abu-Jassar, A. (2025). MICROCHIP MARKING RECOGNITION AND IDENTIFICATION USING A COMPUTER VISION SYSTEM MATHEMATICAL MODEL. *Multidisciplinary Journal of Science and Technology*, *5*(4), 321-330.

15. Chala, O., Yevsieiev, V., Maksymova, S., & Abu-Jassar, A. (2025). USING THE HUMAN FACE RECOGNITION METHOD BASED ON THE MOBILENETV2 NEURAL NETWORK IN AUTHENTICATION SYSTEMS. *Multidisciplinary Journal of Science and Technology*, *5*(3), 882-895.

16. Невлюдов, І. Ш., Євсєєв, В. В., & Гурін, Д. В. (2025). MODEL DEVELOPMENT OF DYNAMIC REPRESENTATION A MODEL DESCRIPTION PARAMETERS FOR THE ENVIRONMENT OF A COLLABORATIVE ROBOT MANIPULATOR WITHIN THE INDUSTRY 5.0 FRAMEWORK. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *1*(79), 42-48.

17. Abu-Jassar, A. T., Attar, H., Amer, A., Lyashenko, V., Yevsieiev, V., & Solyman, A. (2025). Development and Investigation of Vision System for a Small-Sized Mobile Humanoid Robot in a Smart Environment. *International Journal of Crowd Science*, *9*(1), 29-43.

18. Yevsieiev, V., Maksymova, S., Alkhalaileh, A., & Gurin, D. (2025). Development of a program for processing 3d models of objects in a collaborative robot workspace using an HD camera. *ACUMEN: International journal of multidisciplinary research*, *2*(1), 194-210.

19. Yevsieiev, V., Abu-Jassar, A., Maksymova, S., & Demska, N. (2025). Development of a model for recognizing various objects and tools in a collaborative robot workspace. *ACUMEN: International journal of multidisciplinary research*, *2*(1), 224-239.

Scientific adviser: *Vladyslav Yevsieiev, professor of the Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of Radio Electronics.*