Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Research Article

Controlling the Mobile Robot with the Pure Pursuit Algorithm to Tracking the Reference Path Sent from the Android Device

Ahmet TOP[1*]

[1] Fırat University, Faculty of Technology, Electrical-Electronics Engineering Department, atop@firat.edu.tr, Orcid No: 0000-0001-6672-2119

ARTICLE INFO

Article history:

Received 4 January2024
Received in revised form 26 February 2024
Accepted 27 February 2024
Available online 29 March 2024

Keywords:

Communication, Mobile robot, Android application, Pure pursuit algorithm

Doi: 10.24012/dumf.1414768

* Corresponding author

ABSTRACT

Many of the equipment and machines we use in our everyday lives have changed due to major advancements in today's technology. Smartphones, which have made great progress especially in the last decade, perform many tasks in addition to interpersonal communication. Controlling robots, which are increasingly used in daily life and widely included in the literature, is one of these tasks. In this study, the pure pursuit algorithm was used to control the position of a non-holonomic differential drive mobile robot, and the path information to be tracked was received from an Android mobile device as a reference. An application design has been carried out for Android devices. The information for the path drawn here was transferred via the internet to a Google Spreadsheet. Coordinate information obtained from Google tables in MATLAB was separated as x and y axis information and entered into MATLAB/Simulink as waypoints of the pure pursuit algorithm and the position control of the robot was carried out. Error analysis was made by taking the differences between the reference path and the actual movement and the control performance was examined. Additionally, the effect of the approach distance value of the pure pursuit algorithm on the error is presented.

Introduction

Mobile robots have been used effectively in the last decade to perform important tasks in many fields, including military, industrial, and security environments [1,2]. In recent years, as more and more application areas have been opened for robots, production efficiency has increased, manpower has decreased and working environments have improved [3]. There are many types of robots and their tasks. Automating operations like material handling in warehouses, luggage collecting at airports, and mobile security inspection robots are common uses for ground robots. Underwater robots are commonly used to perform sampling, testing, installation, maintenance, and overhaul of groundwater environments, marine environments, and lake and river water environments. Aerial robots are often used to perform aerial search and rescue, search terrain data collection, airborne remote sensing, and other tasks [4,5]. Among these, wheeled mobile robots are used in most applications due to their many advantages such as being fast, having high accuracy, and performing repetitive and difficult tasks easily. These robots are divided into holonomic and non-holonomic wheeled mobile robots. While non-holonomic robots have 2 degrees of freedom (DOF) linear movement in the x-axis and rotational movement in the z-axis, holonomic robots have 3 degrees of freedom because they can also move in the y-axis [6]. Holonomic robots are also called mechanum and omni-wheel. Although these seem more advantageous, their disadvantages are that they are expensive, slow, and slippage [7]. Non-holonomic robots are generally used with differential drive and are called differential drive wheeled mobile robots (DDWMR). DDWMRs can be designed as 2-wheel [8], 3-wheel [9], 4-wheel [10] or 6-wheel [11]. 3-wheeled DDWMRs, which can be controlled more easily thanks to their high maneuverability, are widely used. It consists of two motorized wheels and one caster wheel [12].

One of the most fundamental issues that need to be solved for mobile robots to move and explore on their own in complex environments is path planning [13]. The mobile robot searches for an optimal or suboptimal path from the initial state to the goal state based on certain performance criteria. This is known as the path planning problem [14]. When used properly, path planning techniques for mobile robots can save wear and tear as well as capital costs and save a lot of time. Therefore, the correct choice of navigation technique is the most important step in robot

path planning [15]. There are many position control algorithms such as improved Monte Carlo, pure pursuit, Markov, and Kalman filtering [16]. Of these, the pure pursuit algorithm (PPA) is one of the first. In essence, it is an algorithm that moves to a predetermined point in the distance, concentrates on it, and then approximates its trajectory [17].

In addition to robot design and control, information exchange and control parameters must be sent between the robot and the user. In simulation studies, this process is done directly through the program. However, in practical studies, it can be sent remotely via a computer or a console. However, in this case, since additional hardware is needed, it would be more appropriate to use Android applications in terms of both reducing cost and ease of use. Billions of people around the world use smartphones and this number is increasing rapidly every day. For this reason, mobile applications are preferred for remote access in terms of ease of development. With a single application on a device, other devices can be efficiently managed and monitored [18]. The majority of these devices used are devices with the Android operating system. MIT App Inventor, which enables the creation of application software for Android systems, is an application originally provided by Google and now maintained by the Massachusetts Institute of Technology (MIT). It uses a graphical interface very similar to a block-based visual programming language and allows users to drag and drop visual objects to create an application that can run on the Android system [19]. Robot control with Android applications has received a lot of attention in the literature since Android phones and robot studies have become more and more important in today's world. Aktas et al. [20], in their study, controlled the mobile robot they created with a 3D printer using Wi-Fi and Bluetooth communication with an Android application. Fahmidur et al. [21] controlled the robot via Bluetooth by mirroring the Android phone screen to the computer with the Mobizen application. Sıngporn and Kamon [22] used the MIT App Inventor platform to create a mobile application interface that controls a line-following delivery robot via Bluetooth. Bingöl et al. [23] conducted a study on Bluetooth-controlled wheelchair control with an Android device to facilitate the lives of disabled people. Saravanan et al. [24] studied robot control with voice using Arduino and Android platforms.

In this study, the pure pursuit algorithm was used to control the 3-wheeled DDWMR's position. In contrast to the research in the literature, the algorithm input is provided with a path rather than a few coordinate notices where the robot is given directions to move. An Android application that was developed was used to carry out this path. The application interface has a coordinate system on which the user can design the path they want the robot to take. They can choose three different colors and the appropriate thickness for their path. He/she can also snap a photo of the area wishes to to go by turning on the camera, and then use that image to design a reference path. The x and y coordinate values of the created reference are stored for a certain period with the Android application and when the send button is pressed, they are automatically transferred to the Excel table in Google Spreadsheets, thanks to the easy sharing and real-time editing feature [25]. Arrays were created for x and y coordinates with these values taken from Spreadsheets in the MATLAB program, and these values were applied to the PPA as a waypoint in the control simulation created in MATLAB/Simulink and the robot was controlled. Position controls were provided by sending different paths, and error analyses were also performed by repeating the examinations between the reference path sent from the device and the actual movement for different lookahead values.

Differential drive wheeled mobile robot

In this study, a 3-wheeled mobile robot was used. While there is a standard wheel connected to the right and left motors, there is a caster wheel on the front of the robot that can rotate freely and ensure the balance of the robot. Since it is a non-holonomic robot, it has 2 DOF. For this reason, it can move linearly in the x-axis and rotationally in the z-axis according to the {R} coordinate, as seen in Figure 1. The linear speeds of the right and left motors are V_R and V_L, respectively, and they depend on the radius (r) and angular speed of the wheels, as shown in Equation 1.

$$V_R = r.w_R \quad , \quad V_L = r.w_L \tag{1}$$

The linear speed of the robot is the average speed of the speeds in the x and y axes. However, since the speed of the robot on the y-axis is zero, that is, there is no lateral slip, the average of the right and left linear speeds gives the linear speed of the robot as in Equation 2.

$$V = V_x = \frac{V_R + V_L}{2} = \frac{r}{2}(w_R + w_L) \tag{2}$$

The rotation of the robot occurs in a semicircle according to the effect of the linear speed on the wheels on the center of gravity of the mobile robot. When moving on the circle, clockwise is negative, and counterclockwise is positive. In this case, the angular speed of the robot is calculated according to Equation 3.

$$w = \frac{V_R - V_L}{L} = \frac{r}{L}(w_R - w_L) \tag{3}$$

If the angular velocities of the wheels are distinguished from the equations created here, the inverse kinematic equations in Equations 4 and 5 are obtained.

$$w_L = \frac{1}{r}(V - \frac{wL}{2}) \quad (4)$$

$$w_R = \frac{1}{r}(V + \frac{wL}{2}) \quad (5)$$

where w is the angular speed of the robot (rad/sec), V is the linear speed of the robot (m/s), L is the distance between the two wheels (m), and w_L and w_R are the angular speeds of the left and right wheels (rad/sec), respectively.

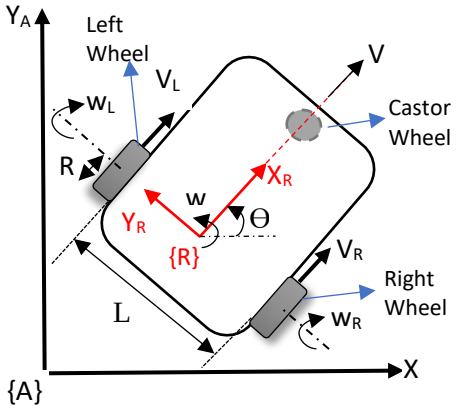


Figure 1. DDWMR motion axis

Looking at the forward kinematic equations in Equations 2 and 3, it is necessary to adjust the angular velocities of the wheels to control the linear and angular velocities of the robot. If the motors to which the wheels are connected are adjusted in the opposite direction and at the same speed, the robot goes forward or backward. If it is turned in the same direction, the robot moves by turning right or left.

Pure pursuit algorithm

The main purpose of position control algorithms is to ensure that the mobile robot moves without deviating from the path. There are many predictive, probabilistic, and geometric-based algorithms developed for this purpose. PPA is a geometric-based algorithm developed in the 1980s for this purpose [26]. The basic logic of the PPA is to determine a lookahead at which it will move, as in humans, and to move by adjusting its speed and orientation to the target point according to its location. It consists of two inputs, the robot's position and target position, and two outputs, the robot's linear and angular speed. As seen in Figure 2, the shortest distance is calculated according to the current position of the robot x_r, y_r, and the target points given to the robot, x_a and y_a, as in Equation 6. After the angle between the robot's position and the target is calculated with Equation 7, the algorithm determines at what linear and angular speeds the robot will move at its output. These speed values are converted into angular velocities for the motors with inverse kinematic equations and this information is sent to the motors. It reaches the final point by updating this information during movement [27].

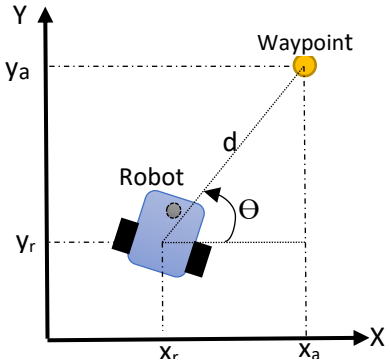


Figure 2. Pure Pursuit approach

$$d = \sqrt{(x_r - x_a)^2 + (y_r - y_a)^2} \quad (6)$$

$$\theta = atan2\big((y_a - y_r), (x_a - x_r)\big) \quad (7)$$

where d is the closest distance between two points, and theta angle represents the angle between two points. The lookahead parameter has an important place for calculations close to the trajectory. When the lookahead parameter is selected large, a wider-angle, smoother, and less oscillating path is followed, as shown in Figure 3 (a). However, in this case, since there will be a lot of deviation from the trajectory in sharp turns, the movement route will be longer, and undesirable long-distance advances will occur. When the lookahead parameter is selected small, oscillations will occur as maneuvers will be made to each viewpoint, as shown in Figure 3 (b). Increasing the oscillation will cause the motors to make sudden speed changes. For these reasons, choosing the lookahead parameter correctly for the trajectories used will increase performance [28].

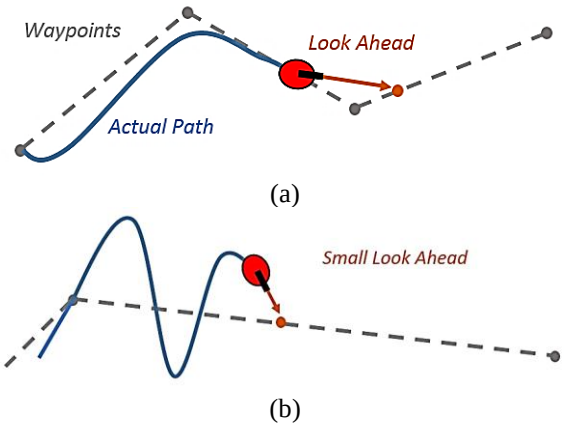


Figure 3. Use of the Lookahead parameter a) large lookahead, b) small lookahead

Android application

An Android-based application has been developed to send the reference path that the robot will follow. The application's main screen is shown in Figure 4 (a). An area of 10 m was created for four regions of the coordinate plane. Here, the user has the option of by hand or by using a stylus to draw the course he/she wishes the robot to go. Red, blue, and green colors can be selected with the three color buttons located under the drawing area. When the camera button is pressed, the camera is turned on and the photo taken is displayed in the drawing area. In this case, the user can draw a path on the photograph taken as in Figure 4 (b). Additionally, line thickness can be increased or decreased with the expansion and reduction buttons. The drawing screen can also be cleared with the clear button. The information on the drawn path is kept as a list in the background, as seen in the instant x-y coordinate information window above. When the send button at the bottom is pressed, this list is transferred to the file created in Google Spreadsheets with information notes in Figure 4 (c).

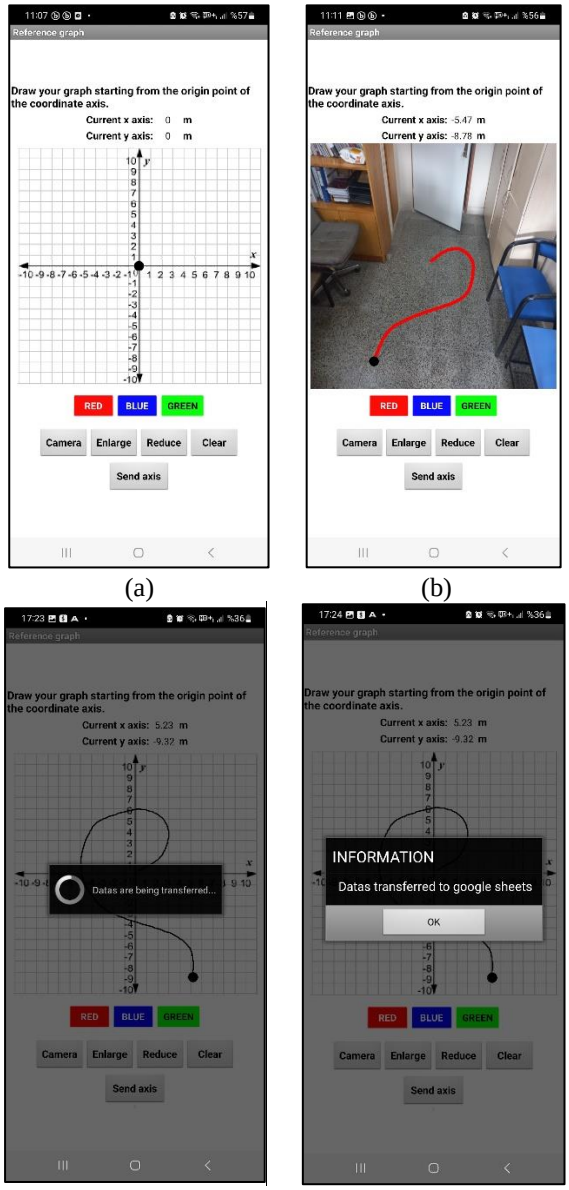


Figure 4. Android application main screen a) drawing on the coordinate axis, b) drawing on the camera image c) information share notes

As shown in the appendices, the application software can be separated into three main parts. Those that follow:

1. Initial variables and settings are made in this section. This is where lists, main screen settings, and axis information beginning values are configured.
2. The part where the button and its operations take place. This is the section where the operations that will take place when the buttons on the main screen are pressed.
3. This is the part used for situations that occur at the time of drawing. It contains the actions that will occur when the ball moves in the drawing area.

Acquisition of data and modeling of control

There are two steps to this section of the study. As illustrated in the workflow diagram in Figure 5, the first section involves obtaining the data from Google Spreadsheets, assigning reference values to arrays, and graphing the Simulink data. Using the provided coordinate data and the simulation shown in Figure 6, the second phase involves using the PPA.

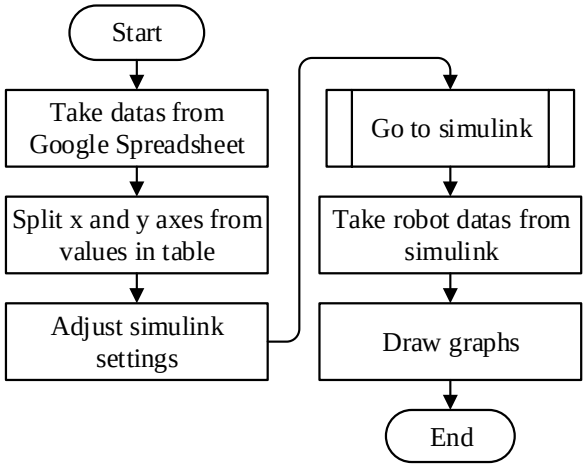


Figure 5. The process of visualizing the outcomes after importing data from Google Spreadsheets

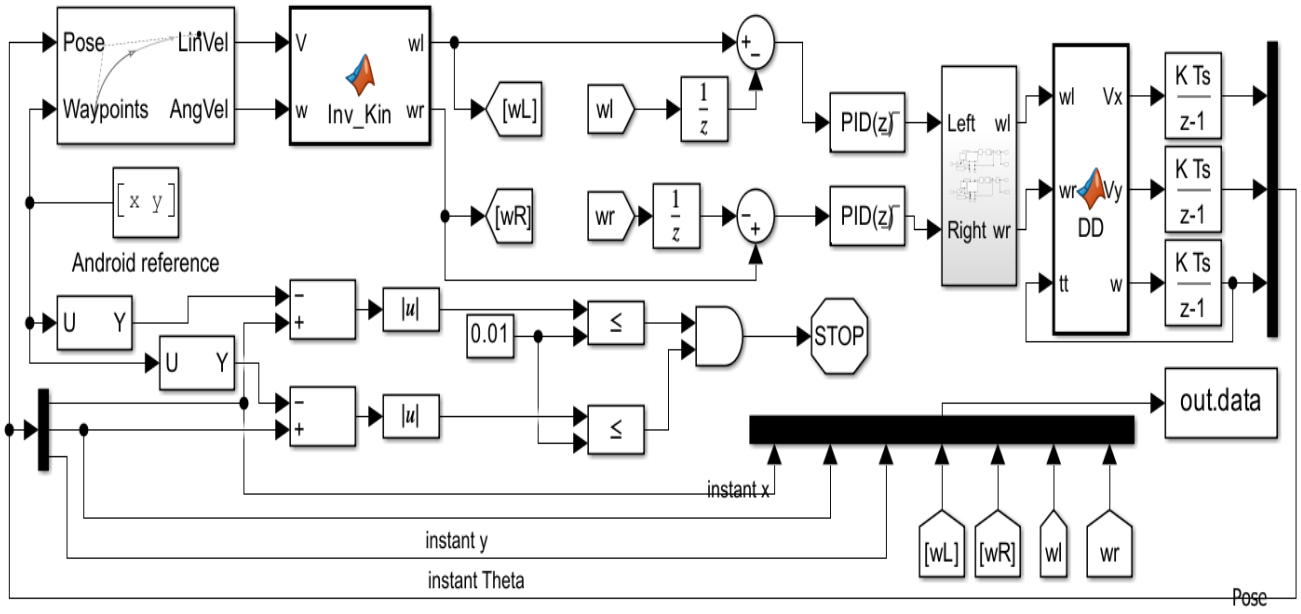


Figure 6. Robot control blocks in Simulink

Instant pose information taken from the robot and reference points taken from the m-file were given as input to the PPA, and the reference linear and angular velocities of the robot were taken from the output. These values were converted into right and left motor reference speeds with inverse kinematic equations, and the motors whose blocks are given in Figure 7 were controlled with the PID controller. The instantaneous speed information of the motors was applied as input to the DD block and was converted back into robot linear and angular speed with the forward kinematic equations in Equations 2 and 3. By multiplying these velocities with the rotation matrix in Equation 8, the velocities in the x and y directions and the angular velocity were found. At the DD block output, position and angle information was obtained by taking their integrals. Since the robot is 2-DOF, the speed in the y direction is zero.

$$Pose = \int \begin{bmatrix} Cos(tt) & -Sin(tt) & 0 \\ Sin(tt) & Cos(tt) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} V_x \\ 0 \\ w \end{bmatrix} \quad (8)$$

By comparing the last reference information and location information, the simulation was stopped with the stop block when the desired range was reached. Robot information was collected via the “to workspace” block and transferred to m-file and graphs of the data were drawn.

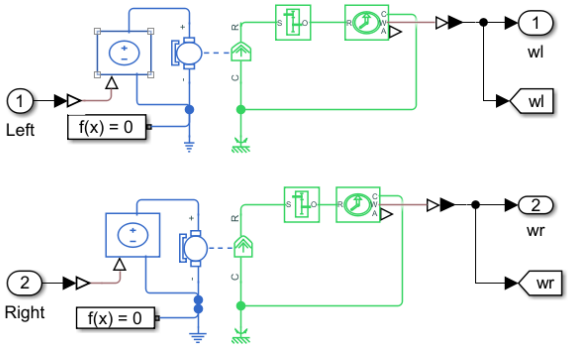


Figure 7: Motor blocks

Implementation of Android application and PPA

Figure 8 illustrates the system's general functioning concept. Through Wi-Fi, the data from the path painted on the Android device was sent to Google Spreadsheets. These data were parsed into x and y arrays after being imported into MATLAB via Wi-Fi using a program written in an m-file.

The created coordinate data was transferred to the Simulink program and applied as input data to the pure pursuit algorithm. When the location control was completed, the data was transferred back to the m-file, and graphs were drawn. Android device and MATLAB outputs for different reference paths are presented in Figure 9.

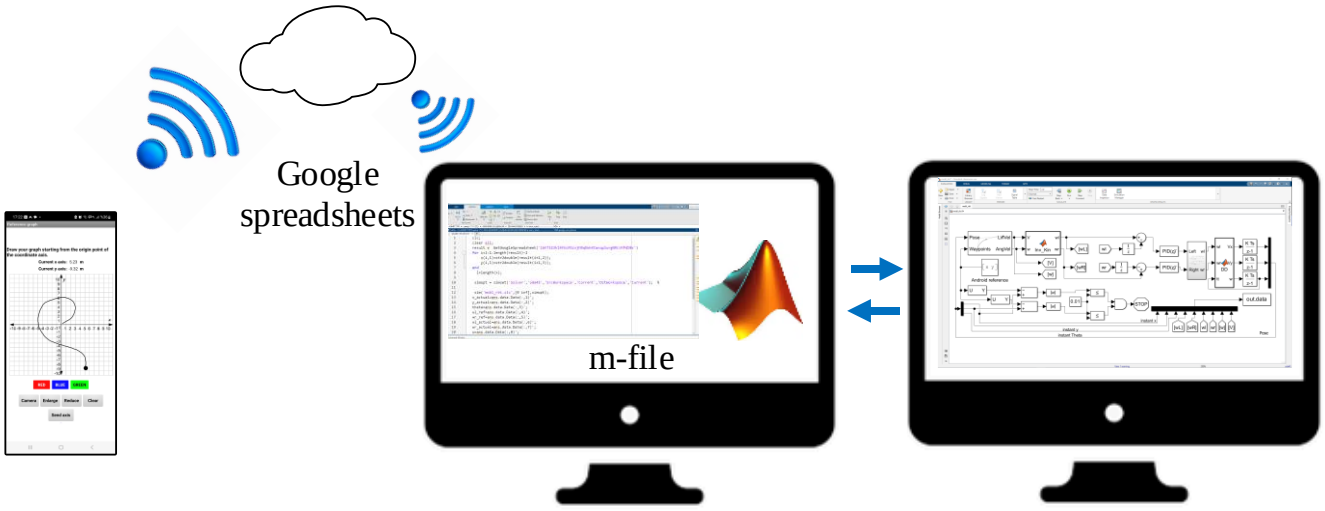


Figure 8. Block diagram of the connection between MATLAB and Android device

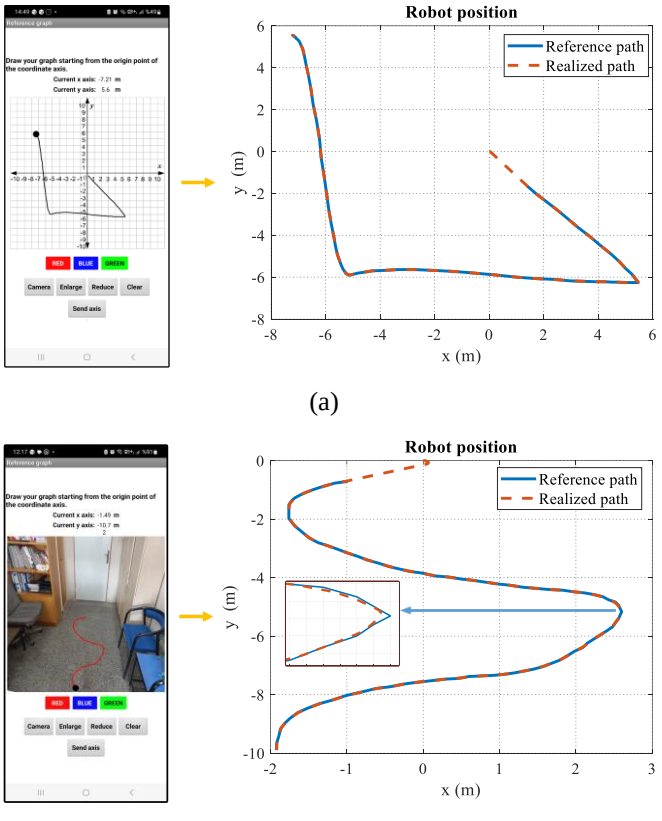


Figure 9. Results for different reference paths a) In the coordinate system and b) in the camera view

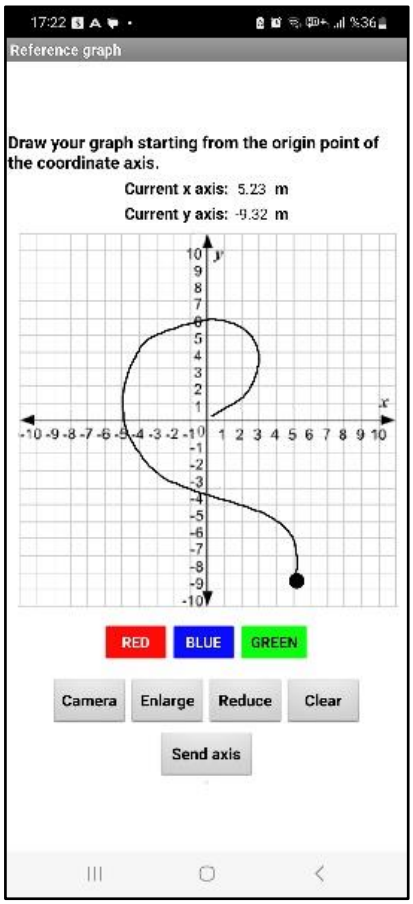


Figure 10. Reference path sent for robot position control

Using the reference path shown in Figure 10, position control was carried out for a maximum of 3 rad/s angular speed, 0.2 m/s linear speed, and 0.1 m lookahead distance. The results, which indicate the robot position, position error, and wheel speeds in Figure 11, were obtained.

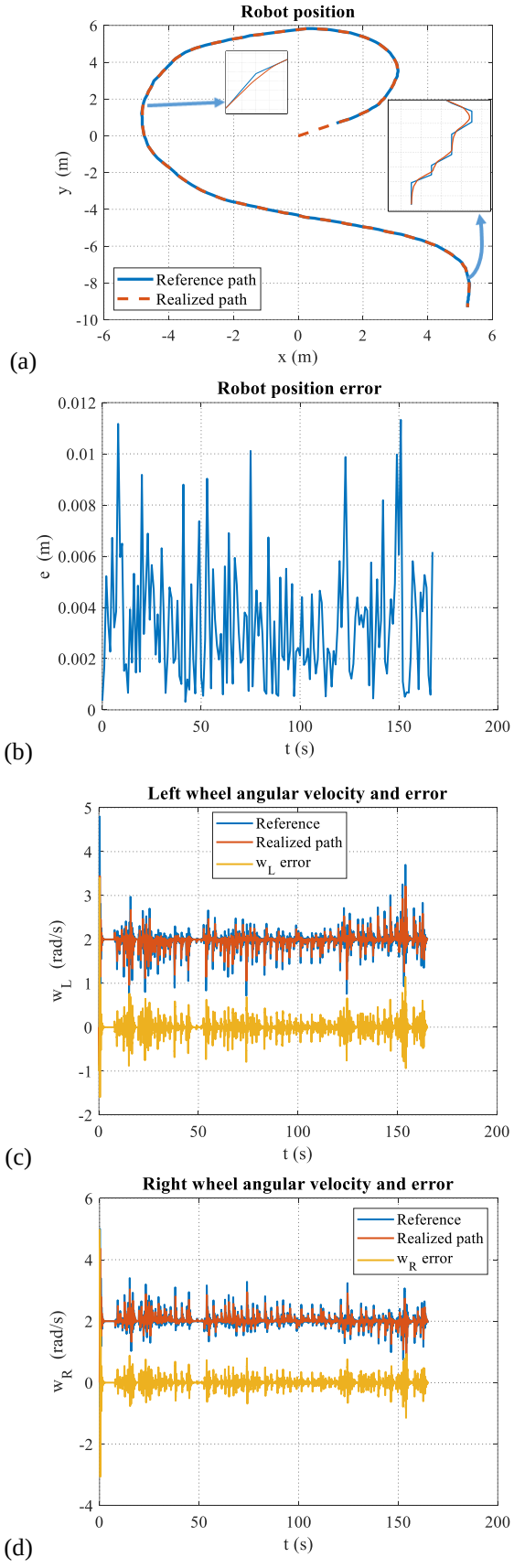


Figure 11. DDWMR position control results: a) path tracking, b) Path tracking error, c) left wheel velocity, d) right wheel velocity

Analyzing the data shows that the robot followed the reference path with a mean error of 0.0034 m and a maximum error of 0.0113 m. The robot's left wheel followed the reference with an angular velocity error of 1.14 rad/s and its right wheel followed the reference with an angular velocity error of 1.7 rad/s for the remaining states after the inertia from the initial movement was eliminated. Figures 12 and 13 show the results for various lookahead distances and the same reference road.

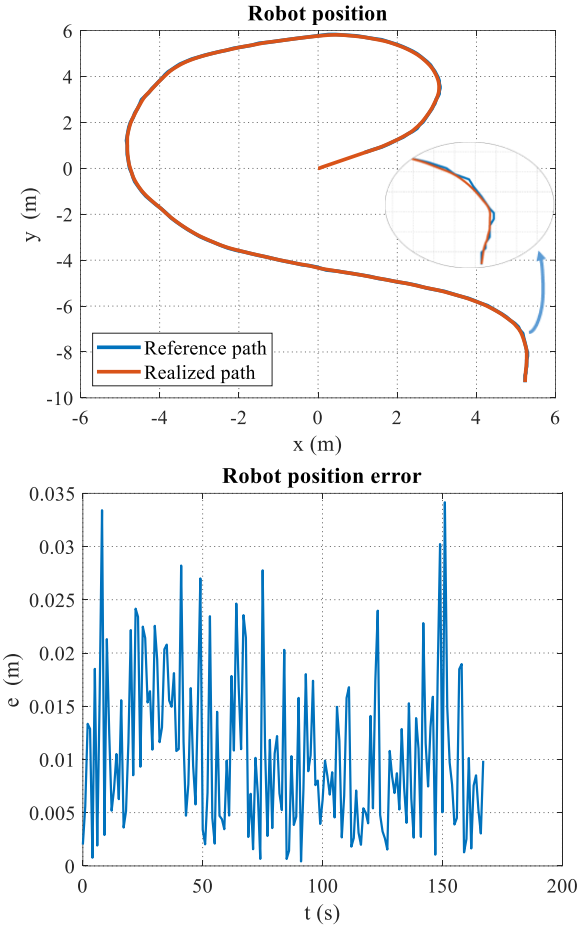


Figure 12. Position control of DDWMR for Lookahead=0.3 m

Table 1. Robot position errors according to different lookahead distances

Lookahead (m)	Maximum error (m)	Mean error (m)
0.1	0.011	0.0034
0.3	0.0113	0.010
0.5	0.0644	0.0243
0.75	0.1118	0.0513
1	0.1759	0.0886

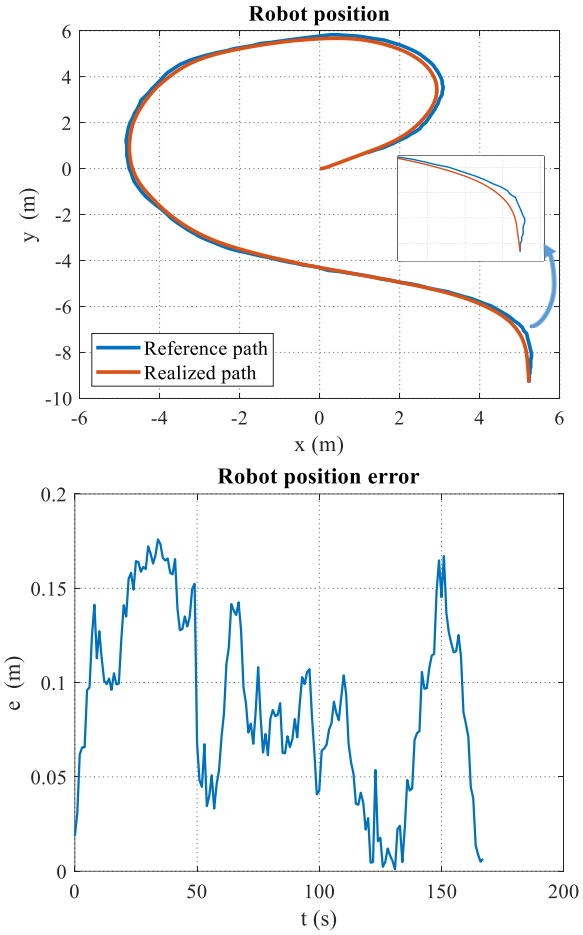


Figure 13. Position control of DDWMR for Lookahead=1 m

The largest difference between the reference path and the actual path was 0.034 m, and the mean error was 0.010 m when the lookahead distance was set to 0.3 m. The maximum and mean errors for the identical scenario, where the lookahead distance is established for 0.5 m, 0.75 m, and 1 m values, are shown in Table 1.

As can be seen from the table, as the lookahead value increases, the error values between the reference and actual position values also increase since the robot follows the reference values from a farther distance.

Conclusion

In the present study, the pure pursuit algorithm was used to control the position of a non-holonomic robot, and an Android application was created to ascertain the reference path. The reference path that the robot wishes to follow in the application created with MIT App Inventor is drawn on the coordinate axis in the drawing area or on an image that can be captured by the camera. Every 500 ms, the path's x and y coordinate values were obtained and added to an array. These coordinate data were transferred via Wi-Fi to the previously established and configured Google Spreadsheet when the application's send button was activated. With the aid of code, these values which became available in Google Spreadsheets over Wi-Fi were moved to a MATLAB m-file, and the axes were split. These reference values were used as waypoints in a MATLAB/Simulink using an m-file to enable pure pursuit to control the robot's location and PID to control its speed. The simulation's output data were imported back into the m-file, where graphs showing the robot's position, velocity, and error values were created. It was observed in the study that the controllers successfully controlled the robot's position and that the data was accurately sent from the Android application. Furthermore, it has been observed that as the lookahead distance increases, the robot's reference tracking error increases. One of the study's advantages is that reference values may be sent to the robot without the requirement for extra equipment. Both the need for extra modules and the Bluetooth short-range reception issue are avoided by using the internet for communication instead of Bluetooth. In addition, when the period time is reduced to get more data from the drawing screen in the Android application, errors occur because values are added to the lists at the same time while drawing. In addition, when drawing slowly to get more data, deviations occur in the data because it detects different points on the finger. For this reason, 500 ms was determined as the optimum time to get more accurate results with a fast drawing. Since the reference coordinates are very close to each other, it moves in constant oscillation as it changes direction at a value very close to the reference at lookahead values below 0.1 m. Therefore, the value of 0.1 m was determined as the lower limit for this study.

Ethics Committee Approval

There is no need to obtain permission from the ethics committee for the article prepared.

Conflict of Interest Statement

There is no conflict of interest with any person/institution in the article prepared.

References

[1] B. Tang, Z. Zhu, and J. Luo, “Hybridizing Particle Swarm Optimization and Differential Evolution for the Mobile Robot Global Path Planning,” *International Journal of Advanced Robotic Systems*, vol. 13, no. 3, p. 86, Jan. 2016, doi: https://doi.org/10.5772/63812.

[2] M. N. A. Wahab, S. Nefti-Meziani, and A. Atyabi, “A comparative review on mobile robot path planning: Classical or meta-heuristic methods” *Annual Reviews in Control*, Oct. 2020, doi: https://doi.org/ 10.1016/ j.arcontrol .2020 .10.001.

[3] S. Lin, A. Liu, J. Wang, and X. Kong, “A Review of Path-Planning Approaches for Multiple Mobile Robots,” *Machines*, vol. 10, no. 9, p. 773, Sep. 2022, doi: https://doi.org/10.3390/machines10090773.

[4] F. Gul, I. Mir, L. Abualigah, P. Sumari, and A. Forestiero, “A Consolidated Review of Path Planning and Optimization Techniques: Technical Perspectives and Future Directions,” *Electronics*, vol. 10, no. 18, p. 2250, Sep. 2021, doi: https://doi.org/10.3390/electronics10182250.

[5] C. Liu, J. Zhao, and N. Sun, “A Review of Collaborative Air-Ground Robots Research,” *Journal of Intelligent and Robotic Systems*, vol. 106, no. 3, Oct. 2022, doi: https://doi.org/10.1007/s10846-022-01756-4.

[6] S. Mellah, G. Graton, E. M. El Adel, M. Ouladsine and A. Planchais, "Actuator Health State Monitoring & Degradation Impact Study on a 4-Mecanum Wheeled Mobile Robot Behaviour," *2021 29th Mediterranean Conference on Control and Automation (MED)*, PUGLIA, Italy, 2021, pp. 1076-1081, doi: 10.1109/MED51440.2021.9480231..

[7] Z. Sun, H. Xie, J. Zheng, Z. Man, and D. He, "Path-following control of Mecanum-wheels omnidirectional mobile robots using nonsingular terminal sliding mode," *Mechanical Systems and Signal Processing*, vol. 147, p. 107128, Jan. 2021, doi: https://doi.org/10.1016/j.ymssp.2020.107128.

[8] R. P. M. Chan, K. A. Stol, and C. R. Halkyard, "Review of modeling and control of two-wheeled robots," *Annual Reviews in Control*, vol. 37, no. 1, pp. 89–103, Apr. 2013, doi: https://doi.org/10.1016/j.arcontrol.2013.03.004.

[9] S. Peng and W. Shi, "Adaptive Fuzzy Output Feedback Control of a Nonholonomic Wheeled Mobile Robot," in *IEEE Access*, vol. 6, pp. 43414-43424, 2018, doi: 10.1109/ACCESS.2018.2862163.

[10] M. Begnini, D. W. Bertol, and N. A. Martins, "A robust adaptive fuzzy variable structure tracking control for the wheeled mobile robot: Simulation and experimental results," *Control Engineering Practice*, vol. 64, pp. 27–43, Jul. 2017, doi: https://doi.org/10.1016/j. conengprac. 2017.04.006..

[11] H. Zhao, C. Luo, Y. Xu, and J. Li, "Differential Steering Control for 6 × 6 Wheel-drive Mobile Robot," *2021 26th International Conference on Automation and Computing (ICAC)*, Portsmouth, United Kingdom, 2021, pp. 1-6, doi: 10.23919/ICAC50006.2021.9594210.

[12] P. Petrov and V. Georgieva, "Adaptive Velocity Control for a Differential Drive Mobile Robot," *2018 20th International Symposium on Electrical Apparatus and Technologies (SIELA)*, Bourgas, Bulgaria, 2018, pp. 1-4, doi: 10.1109/SIELA.2018.8447091.

[13] G. Klančar, A. Zdešar, and M. Krishnan, "Robot Navigation Based on Potential Field and Gradient Obtained by Bilinear Interpolation and a Grid-Based Search," *Sensors*, vol. 22, no. 9, p. 3295, Apr. 2022, doi: https://doi.org/10.3390/s22093295.

[14] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Applied Soft Computing*, vol. 30, pp. 319–328, May 2015, doi: https://doi.org/10.1016/j .asoc. 2015.01.067.

[15] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of Autonomous Path Planning Algorithms for Mobile Robots,", *Drones*, vol. 7, no. 3, pp. 211–211, 2023, doi:https://doi.org/10.3390/drones7030 21 1.

[16] S. K., Malu, & J. Majumdar,. "Kinematics, localization and control of differential drive mobile robot". *Global Journal of Research In Engineering*, *14*(1), 1-9. 2014

[17] M. Samuel, M. Maziah, M. Hussien, and N. Y. Godi, "Control of Autonomous Vehicle Using Path Tracking: A Review," *Advanced Science Letters*, vol. 24, no. 6, pp. 3877–3879, Jun. 2018, doi: https://doi.org/10.1166/ asl. 2018.11502.

[18] S. Hong, "An Efficıent Iot Applıcatıon Development Based On Iot Knowledge Modules," *Issues In Information Systems*, 2020, doi: https://doi.org/10.48009/3_iis_2020_72-82.

[19] E. Pasternak, R. Fenichel, and A. N. Marshall, "Tips for creating a block language with blockly," *2017 IEEE Blocks and Beyond Workshop (B&B)*, Raleigh, NC, USA, 2017, pp. 21-24, doi: 10.1109/BLOCKS.2017.8120404.

[20] M. Aktaş, F. Polat, and M. Oflezer, "Bluetooth Ve Wifi Kontrollü Mobil Robot Tasarımı Ve Uygulaması", *İleri Teknoloji Bilimleri Dergisi*, vol. 7, no. 3, pp. 29–35, 2018.

[21] R. K. Fahmidur, H. M. A. Munaim, S. M. Tanvir and A. S. Sayem, "Internet controlled robot: A simple approach," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, 2016, pp. 1190-1194, doi: 10.1109/ICEEOT.2016.7754873.

[22] P., Sıngporn, & S. Kamon, "Controlling the Line Follower Delivery Robot with MIT APP Inventor". *Journal of Technology and Innovation in Tertiary Education*, *1*(1), 9-16, 2018

[23] O. Bingöl, Ö. Aydoğan, B. Özkaya, N. Şen, "Android Cihaz ile Tekerlekli Sandalye Kontrolü", *Afyon Kocatepe Üniversitesi Fen ve Mühendislik Bilimleri Dergisi, Özel Sayı* (164-169). 2016

[24] M. Saravanan, B. Selvababu, A. Jayan, A. Anand, and A. Raj, "Arduino Based Voice Controlled Robot Vehicle," *IOP Conference Series: Materials Science and Engineering*, vol. 993, p. 012125, Dec. 2020, doi: https://doi.org/10.1088/1757-899x/993/1/012125.

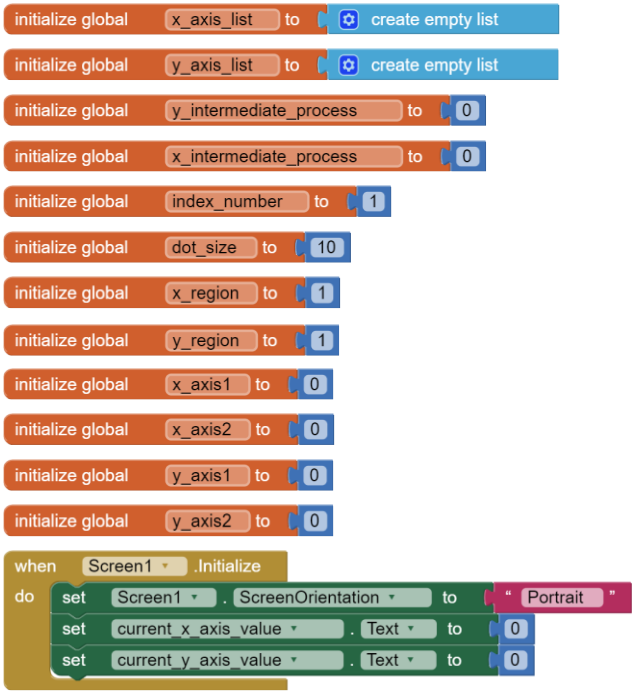
[25] Google Sheets features, Access Date: 01 January 2024 https://www.google.com/sheets/about/

[26] M. Samuel, M. Maziah, M. Hussien, and N. Y. Godi, "Control of Autonomous Vehicle Using Path Tracking: A Review," *Advanced Science Letters*, vol. 24, no. 6, pp. 3877–3879, Jun. 2018, doi: https://doi.org/10.1166/asl. 2018.11502.

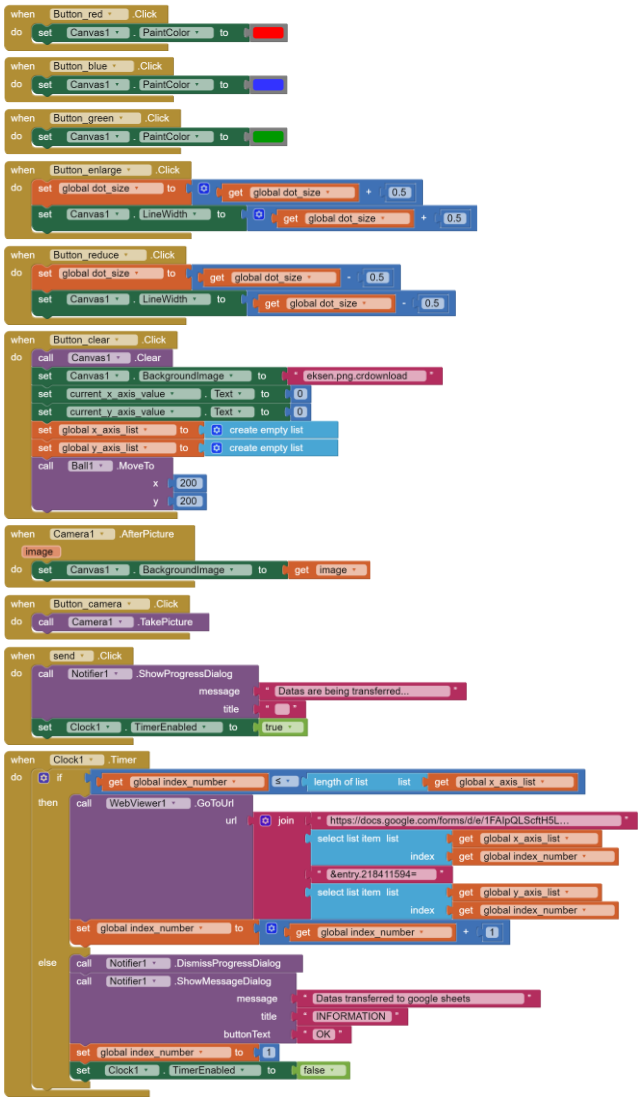
[27] G. GÜRGÜZE, & İ. TÜRKOĞLU, "Dinamik Modeli Bilinen Diferansiyel Mobil Robotun Pure Pursuit Algoritması İle Pozisyon Kontrolünün Yapılması", International Congress on HumanComputer Interaction, Optimization and Robotic Applications, 2019

[28] -J. Wang, T. -M. Hsu and T. -S. Wu, " The improved pure pursuit algorithm for autonomous driving advanced system" *2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA)*, Hiroshima, Japan, 2017, pp. 33-38, doi: 10.1109/IWCIA.2017.8203557.

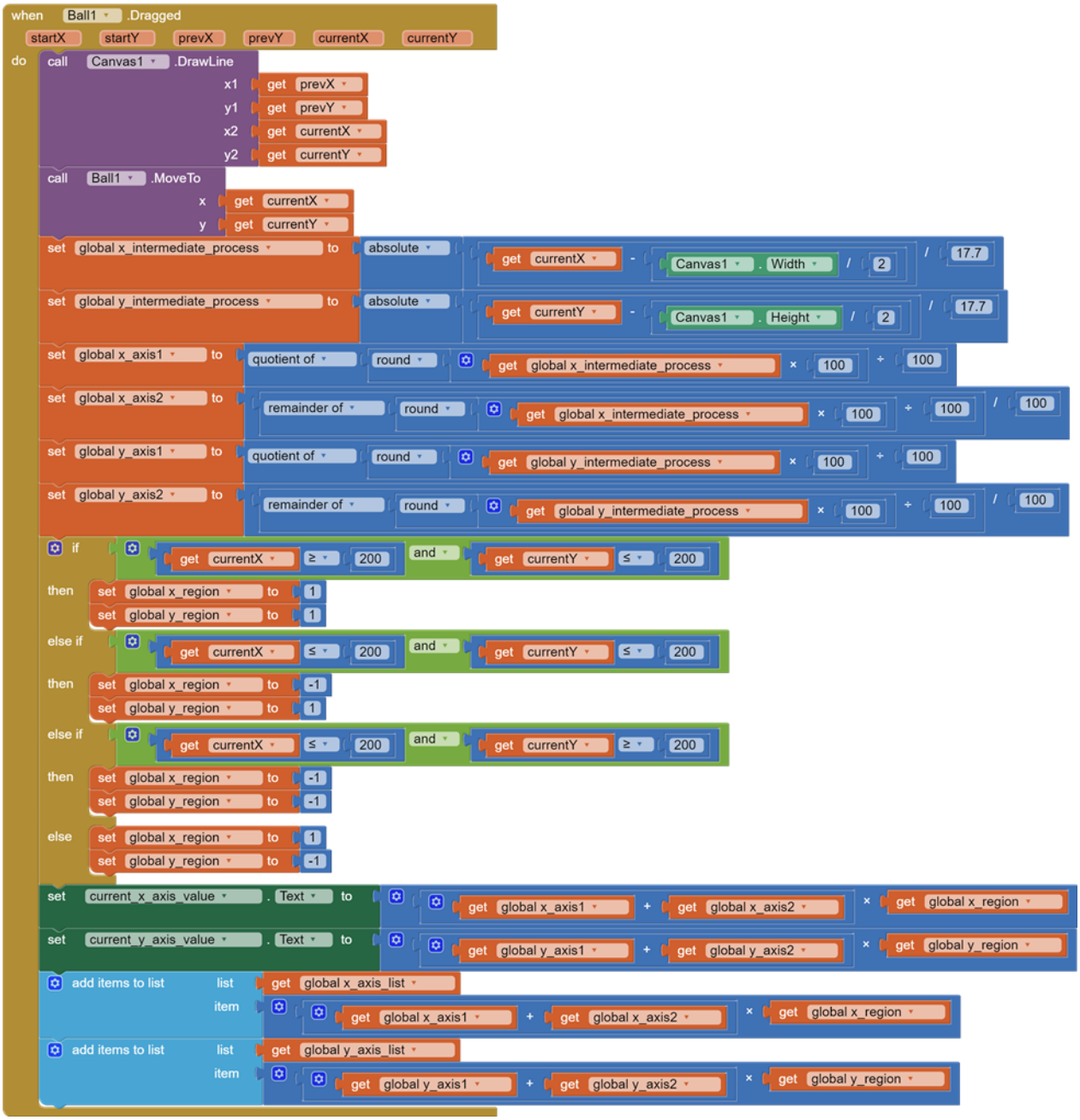
Appendix


initialize global x_axis_list to create empty list
initialize global y_axis_list to create empty list
initialize global y_intermediate_process to 0
initialize global x_intermediate_process to 0
initialize global index_number to 1
initialize global dot_size to 10
initialize global x_region to 1
initialize global y_region to 1
initialize global x_axis1 to 0
initialize global x_axis2 to 0
initialize global y_axis1 to 0
initialize global y_axis2 to 0
when Screen1 .Initialize
do set Screen1 . ScreenOrientation to " Portrait "
set current_x_axis_value . Text to 0
set current_y_axis_value . Text to 0

(a)


when Button_red .Click
do set Canvas1 . PaintColor to
when Button_blue .Click
do set Canvas1 . PaintColor to
when Button_green .Click
do set Canvas1 . PaintColor to
when Button_enlarge .Click
do set global dot_size to get global dot_size + 0.5
set Canvas1 . LineWidth to get global dot_size + 0.5
when Button_reduce .Click
do set global dot_size to get global dot_size - 0.5
set Canvas1 . LineWidth to get global dot_size - 0.5
when Button_clear .Click
do call Canvas1 .Clear
set Canvas1 . BackgroundImage to " eksen.png.crdownload "
set current_x_axis_value . Text to 0
set current_y_axis_value . Text to 0
set global x_axis_list to create empty list
set global y_axis_list to create empty list
call Ball1 .MoveTo
x 200
y 200
when Camera1 .AfterPicture
image
do set Canvas1 . BackgroundImage to get image
when Button_camera .Click
do call Camera1 .TakePicture
when send .Click
do call Notifier1 .ShowProgressDialog
message " Datas are being transferred... "
title " "
set Clock1 . TimerEnabled to true
when Clock1 .Timer
do if get global index_number ≤ length of list list get global x_axis_list
then call WebViewer1 .GoToUrl
url join " https://docs.google.com/forms/d/e/1FAIpQLScfH5L...
select list item list get global x_axis_list
index get global index_number
" &entry.218411594= "
select list item list get global y_axis_list
index get global index_number
set global index_number to get global index_number + 1
else call Notifier1 .DismissProgressDialog
call Notifier1 .ShowMessageDialog
message " Datas transferred to google sheets "
title " INFORMATION "
buttonText " OK "
set global index_number to 1
set Clock1 . TimerEnabled to false

(b)



(c)

a) Initial screen settings, b) button functions, and c) drawing area blocks