Design of a Pose Sensor Comprising N Optical Mice for Implementation in an Autonomous Platform Robot

Laura Patiño Restrepo[a], Davinson Castaño Cano[b] (tutor)

[a] *EAFIT University Physics Engineering student, Medellin, Colombia, lpatinor@eafit.edu.co*
[b] *EAFIT University professor, Medellin, Colombia, dcasta25@eafit.edu.co*

ARTICLE INFO

Keywords:
Optical mouse
Odometry
Sensor fusion
Diffrential drive robot
Pose
Bit Banging
ROS

ABSTRACT

This paper addresses the design of an optical pose sensor based on sensor fusion of N optical mice sensors to estimate the pose and real time odometry of an autonomous differential drive robot. This proposal, unlike common wheel encoder based position and pose sensors, does not present *slippage* problem and could be implemented with a low cost budget.This project is developed under the Advanced Project I course from the Physics Engineering program at EAFIT University. It encloses the review of the corresponding mathematical model, the communication protocol with the sensors via bit banging in Arduino, the design of the data processing code in Python and the display interface using a URDF model in ROS2. Several tests were performed to prove effectiveness in specific indoor surfaces and medium precision with an absolute trajectory error (ATE) from 30.95 - 171.25 mm facing the challenges of calibration and simultaneity between mice readings and drift error.

1. Introduction

The fields of robotics and automated ground mobility have been widely in demand in the industry for the last few years with indoor and outdoor applications such as house cleaning technologies, inside factories or food deliveries, surveillance and even space research robots [1]. In this applications, robots need to be able to sense and interact with the environment and their user needs to be able to track them, therefore, they have to use position and pose (position plus orientation) sensors as a block of their closed control loop because it is the retrieved sensed data that allows them to get feedback and make new decisions.

There are typically two ways of sensing the environment through localization: i) absolute such as GPS or by pre-mapping the environment by specific features or ii) relative such as odometry which estimates the robot's displacement incrementally, based on its immediately previous estimated position and the accumulation of movements [2]. Odometry has been usually performed by incremental wheel encoders that measure the amount of wheel rotation in order to know the total displacement, but these often experience the problem of *slippage* or wheel skid that introduces great error in addition to their high costs. Commercial cheap optical mice sensors have a CMOS camera and a digital signal processor (DSP) that allows them to know with good precision the displacements in two dimensions through the optical flow by the relative movement of pixels and the analysis of their intensity patterns [3][4].

However, the use of optical mice in sensor fusion technique to estimate pose and odometry in three dimensions is not new in academic research. Various mathematical models have been proposed based on trigonometric approach by coordinate transformation between frames of reference or by translation and rotation transformation matrices [2][5][6][7].

Regarding the location and number N of sensors that should be used used to perform the correct localization of the robot in a global coordinate frame, the magnitude of the measurement is not affected by the orientation of the sensors in their individual coordinate frame[8], however, if these frames are aligned with the robot's, their mathematical processing becomes simpler [2]. For a differential robot, the sensors should be located as far as possible from the main center of rotation, although some of the sensors should be on the axis where the dynamic center of rotation moves[9] therefore, it could be a triangular arrangement. Additionally, the more sensors used, the more the error is compensated and the better the approximation of the pose, as long as the information between them is coherent[9]. As for this subject, latent problems have been identified such as the synchronization of data collection from the different sensors that cause the accumulation of error (drift error) due to the communication protocols used [2] and quality of different surfaces on which they work[3].

In the following sections, we present the design of a pose sensor based on the previous recommendations to be used in a differential drive robot where the actuator controller and the actuator itself have already been introduced with innovative proposals. We pretend to determine what is the error or effectiveness in implementing optical mouse sensors in pose and odometry retrieval with our configuration and parameter selection. Design of specific control algorithms for obstacle avoidance among others requested by the user and the integration of all the systems would remain to be implemented in future work.

2. Methodology

The methodology used to elaborate this work is based on iterative methodology, where research, design, experimental tests and correction implementation are cross-functional stages in order to achieve rapid functionality initially with two sensors and subsequently with more than two sensors to evaluate the differences and advantages. To accomplish this, the following segments are discussed.

2.1. Communication with sensors

This project uses the FCT3065-XY sensors for Bluetooth optical mice with resolution of 1000 dpi. These sensors do not have a data sheet available for the user, however they are compatible with other models [10]. The sensors are connected in a way where each their VDD are connected to 3.3 V of an Arduino MEGA, their SDIO (data pin) and SCLK (clock pin) are connected to PWM pin pairs or to the specific communication pins (20 and 21), and the grounds must be interconnected between them. The protocol used for communication between the sensors and the Arduino is bit banging based on the approach in a GitHub repository[10], where serial communication is performed entirely through software 1 bit at a time by manipulating pin states (HIGH or LOW and INPUT or OUTPUT) in all the process. Corrections and conditioning for two (as it is shown in flow chart of Figure 1) and three sensors were implemented [11]. From this code, information of position in x and y in mm is retrieved, being classified according to the number of sensor where it belongs.

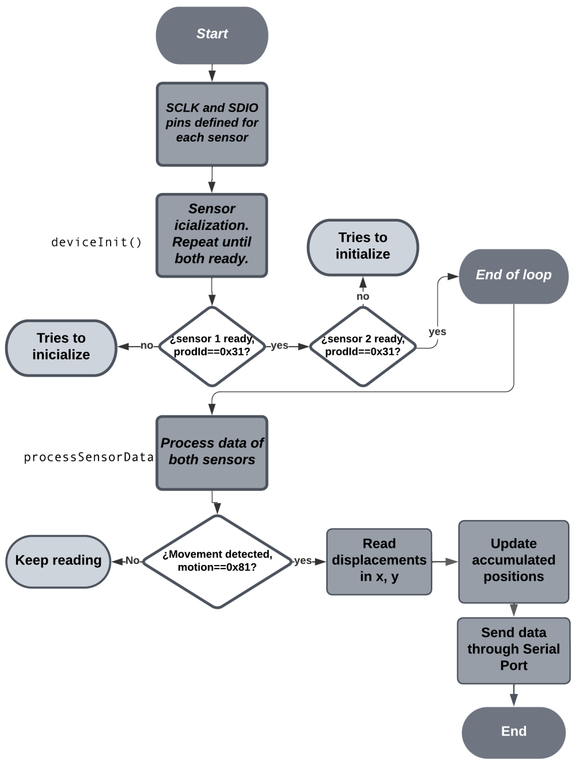
Start
SCLK and SDIO pins defined for each sensor
deviceInit()
Sensor icialization. Repeat until both ready.
Tries to initialize
End of loop
Tries to inicialize
no
¿sensor 1 ready, prodId==0x31?
yes
¿sensor 2 ready, prodId==0x31?
no
yes
processSensorData
Process data of both sensors
Keep reading
No
¿Movement detected, motion==0x81?
yes
Read displacements in x, y
Update accumulated positions
Send data through Serial Port
End

Figure 1: Flow chart: communication with sensors through Bit Banging in Arduino.

2.2. Data Processing and Mathematical Model

To estimate the pose and odometry for every moment t and $t+1$, we need to introduce a mathematical model that is implemented in Python after doing the communication with the serial from Arduino through the library $PySerial$. The experimental set-up used for this study consists initially in two mouse sensors (1 and 2) solidly attached by a distance D, parallel to each other and orthogonal to their rotation axis as it is shown in Figure 2. Consider the fact that this set-up is the robot's simulation, however it is important to remember differential drive robots are non holonomic systems which means they can't move in all of their degrees of freedom, in this case, they can't move laterally in x axis without changing their orientation.

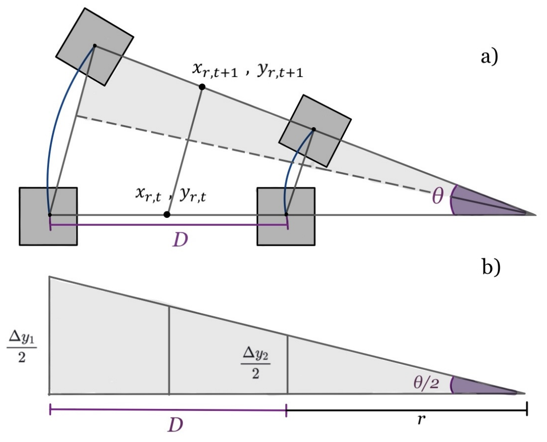

Figure 2: Mathematical model geometry. a) general movement overview b) mathematical trigonometric approximation.

We start from the approximation of considering every position change of the system is in arc movement because of what was explained above, and as this changes are happening too fast, they can also can also be approximated as a straight line trajectory with an x and y displacement. We assume their mid-point as the center of the robot, therefore, the total displacement can be calculated as follows, knowing that the x or y displacements of each sensor are obtained from the subtraction of a current value minus the last value registered.

$$\Delta y_r = \frac{\Delta y_2 + \Delta y_1}{2} \quad , \quad \Delta x_r = \frac{\Delta x_2 + \Delta x_1}{2} \tag{1}$$

Then, given the geometry, we can derive the expression for $\Delta\theta_r$ of the robot starting from the equation system in eq.2

$$\tan\frac{\Delta\theta}{2} = \frac{\Delta y_2}{2r} \quad , \quad \tan\frac{\Delta\theta}{2} = \frac{\Delta y_1}{2(r+D)} \tag{2}$$

$$\Delta\theta_r = 2\tan^{-1}\left(\frac{\Delta y_2 - \Delta y_1}{2D}\right) \tag{3}$$

Finally, $\Delta\theta_r$ is used to compute the angle in present time once all data is updated from present time (x_t, y_t, θ_t) to an immediate past time and immediate last values $(x_{t-1}, y_{t-1}, \theta_{t-1})$. With this new values, the data processing loop is executed again to find the next current values of robot's position in terms of the world coordinate system as shown in eqs.4, 5, 6. For this math to work out, we need to be very careful with all the variables' initialization in the code.

$$\theta_t = \theta_{t-1} + \Delta\theta_r \tag{4}$$

$$x_t = x_{t-1} + \Delta x_r \cos\theta_{t-1} - \Delta y_r \sin\theta_{t-1} \tag{5}$$

$$y_t = y_{t-1} + \Delta x_r \sin\theta_{t-1} + \Delta y_r \cos\theta_{t-1} \tag{6}$$

After this model for two sensors was proven effective, it was escalated to three pairs of sensors in the configuration that was mentioned earlier in section 1 proved be more optimal, changing the respective distances between the two mouses in consideration and then calculating the mean value for the pose with this three values to use the redundancy as an error counteracting.

2.3. *Visualization interface*

To accomplish odometry visualization of the robot in real time in order to facilitate tracking during experimental tests to verify correct operation or for future users tracking, it was decided to design a URDF (Unified Robot Description Format) model based on a node system in Python through ROS2 operating system available on this projects' GitHub [11], where every node is a part of the total process and they communicate between them through messages called topics, a node publishes a topic and other node subscribes to that topic and uses its information.

In Figure 3, we show the node connectivity for our specific goal. Pose Publisher node is the one where the mathematical model shown in section 2.2 is applied with an update time rate of 1 ms, Odom Publisher allows to obtain the incremental pose tracking in vector paths and TF Publisher allows to establish static transformations or static geometric relations between reference frames for example between robot's coordinate frame (placed in their center) and global coordinate frame (initial position) or between the sensors as they are placed with fixed distances; finally, these odometry and transformations are published in RViz2, the display or interface tool specific to ROS2. To date, robots' geometry is shown in a simplified way called "turtle" according to the package *turtlesim*, in future work, the actual CAD model of the robot could be implemented rapidly to this URDF model.

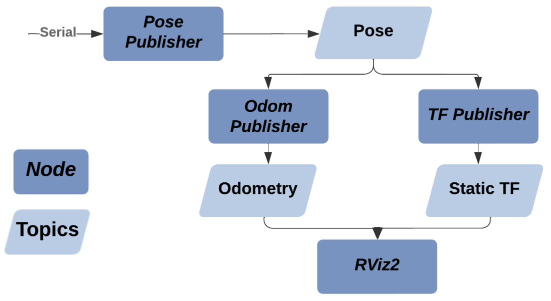

Figure 3: General schematic of ROS nodes connectivity in Python.

It is important to note that RViz 2 only interprets angles from the pose in quaternion form (q_x, q_y, q_z, q_w), thus a transformation is needed and prior to that, angles must be written as Euler angles (ϕ, θ, ψ) which indicates rotations as roll in x axis, pitch along y axis and yaw in z axis accordingly. In this case, a differential drive robot only has rotation along their vertical axis z so we assume $(0, 0, \psi)$, where ψ is actually our θ obtained in the pose, then we proceed to apply the following equations.

$$\alpha_i = \frac{\phi}{2} \qquad \alpha_j = \frac{\theta}{2} \qquad \alpha_k = \frac{\psi}{2} \tag{7}$$

$$\begin{array}{lll} c_i = \cos(\alpha_i) & s_i = \sin(\alpha_i) & c_j = \cos(\alpha_j) \\ s_j = \sin(\alpha_j) & c_k = \cos(\alpha_k) & s_k = \sin(\alpha_k) \end{array} \tag{8}$$

$$\begin{aligned} q_x &= c_j \cdot c_k \cdot s_i - s_j \cdot s_k \cdot c_i \\ q_y &= s_j \cdot c_k \cdot c_i + c_j \cdot s_k \cdot s_i \\ q_z &= c_j \cdot s_k \cdot c_i - s_j \cdot c_k \cdot s_i \\ q_w &= c_j \cdot c_k \cdot c_i + s_j \cdot s_k \cdot s_i \end{aligned} \tag{9}$$

3. Results and Discussion

In order to evaluate the aspects mentioned in the methodology section, experimental tests were carried out manually due to the conditions of the experimental set up in this project stage. First indicators of good performance and application of the mathematical model were observed during a zig-zag trajectory test where at each vertex the system needed to rotate to continue the desired path. The odometry was recorded in Rviz2 as a path formed by red vectors and a yellow straight line indicates the link with the robot's coordinate frame and the starting point in the global coordinate frame (see Figure 4)

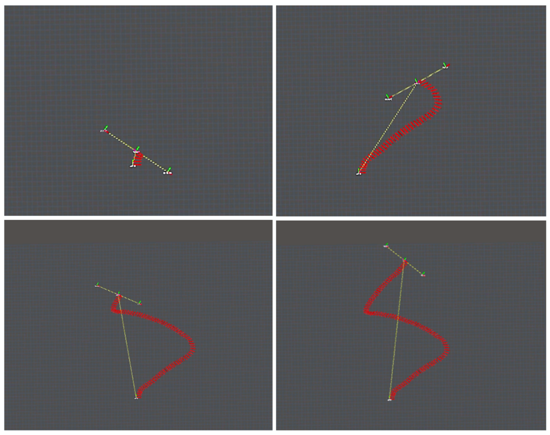

Figure 4: Operation tests: Zig-Zag Trajectory - RViz2 Pose and Odometry Visualization.

After the implementation of sensor fusion in the mathematical model of the three pairs of sensors, other experimental tests were proposed in order to have a better validation of the performance. First experiment consists in a fixed inverted

L trajectory, where both segments measured 300 mm and then again, the system needed to rotate -90° once it reached the vertex to continue the desired path. Data was recorded and stored for off-line analysis in MATLAB. Figure 5 shows the comparison between the position tracking performance of each pair of sensors in the system with the mean value calculated with the average of the three sensors, and also the comparison between these and the real desired or reference trajectory. We can see how they all resemble the trajectory of the inverted L with more difference when approaching the rotation in the vertex as it will be explained further later on.

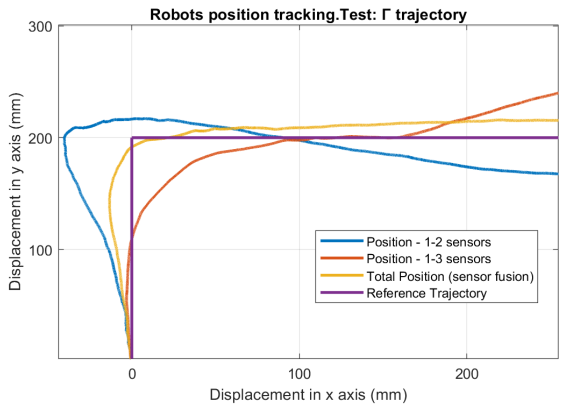

Figure 5: Robot's position tracking - inverted L trajectory. Model comparison.

Then, in Figure 6 we can observe a graph with a complete pose plot showing the evolution of x, y and θ during the sampling time that is not fixed and depends on each individual test.

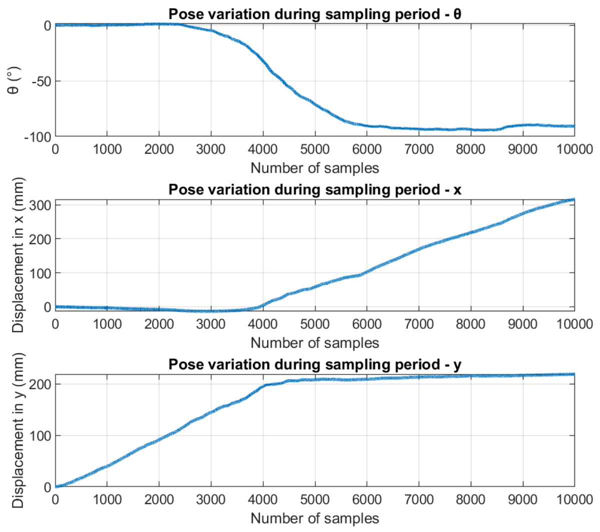

Figure 6: Pose evolution plot - inverted L trajectory.

A common way to quantify the results' accuracy in odometry and robotics research works is the calculation of the ATE (Absolute Trajectory Error) statistic consisting in the determination of how well an estimated trajectory aligns with a reference trajectory (ground-truth trajectory) linearly interpolated point by point as described in the formula of eq. 10 using RMS. Even if this statistic only refers to position deviation in 2D and not pose in 3D, as it was explained in the mathematical model, x and y of the robot frame of reference in the global frame of reference became now dependant of the theta angle, which means, we can also in a way deduce the error of the whole pose model implementation with this statistic.

$$ATE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|p_i - \hat{p}_i\|^2} \tag{10}$$

In Table 1 we present the comparison between ATE values for the different sensor pairs (2 in this case which presented better results due to an alleged decalibration of sensor 2) and the sensor fusion implementation. We can observe as expected from theory and the position tracking plot presented above that the best fitting results are obtained with the sensor fusion as it has a lower ATE which means a smaller total deviation from the reference trajectory. It may seem a big number but taking into account the main application is not high precision robotics but localization of medium size robots, it is considered an acceptable value, the experimental error is also associated with human error of performing the movement alongside the desired path.

Table 1
ATE values for position approximation with different pairs of sensors.

sensor pair	1-2	1-3	sensor fusion
ATE (mm)	39.24	86.07	30.95

We had planned other tests comprising a repeated number of the same arc movement but decided to perform a semi - circular trajectory test (with radius 20 cm) instead as it is easier to establish their ground-truth trajectory for comparison. The results of this test from Figure 7 show that the model acts correctly until the angles surpass 90° rotations and then the drift error increases gradually generating an offset and a total ATE to 171.25 mm, which would need to be corrected adding more robustness to the model. This results are also attached and are sensible to the human error during the performance of the experiments.

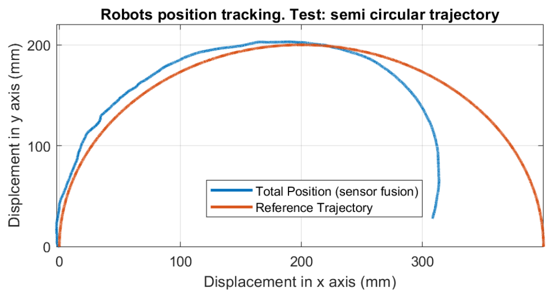

Figure 7: Robot's position tracking - semi circular trajectory.

During the course of this experiments, a main point was identified; the model fails to perform correctly when the system or robot rotates completely on only one of its sensors, meaning the dynamic pivot point is located at one end of the axis , introducing an error to almost 60°, this may occur due to the assumption of the mathematical model of every movement of the system as an arc trajectory. To avoid this error, the system or robot needs to be moving when it applies a rotation (by applying acceleration to one of its wheels).

Another aspect that was identified, was the behavior change in different surfaces, introducing an error a lot greater than 20° (the average maximum error in a straight line along the y axis in a good surface) in some surfaces. This was somewhat expected as we know mice work better in less homogeneous surfaces (but not completely irregular) because the sensor detects features and differences in the surface easier but also they can't be shiny surfaces like a very varnished wood (was tested) because it can cause additional light reflections that interfere with the correct reading in the sensor. This is an important observation because it shows a disadvantage in this mouse sensor implementation in comparison to IMUs, GPS or encoders that do not depend on the surface and could have a wider range of outdoor applications.

Finally, let's take into account that complete failure of the model will be presented when for any reason sensors stop receiving information in a simultaneous way or one sensor fails to read displacements at all, so it is very important to keep this process monitored to not mistakenly believe that they are real results of the model.

4. Conclusions and future work

As a project developed under a college course, it strengthened physical mathematical as well as engineering knowledge and understating such as in programming, electronics and a new overview of the robotics area and its aspects. Throughout its development, several challenges were encountered with software, hardware (like the calibration and reliability of each individual sensor), or systematic and non systematic errors during the experimental tests to prove the model validation; this needs to be taken into account when evaluating the results but this also allowed to seek and learn problem solving and improve organised research and recursion skills which are very important for future projects.

The project did not reach the completion and final stage that was initially proposed however it did fulfill the main objective of estimating the pose of a system simulating an autonomous robotic platform in real time through the development of an optical measurement system with N=3 commercial mouses. Specific goals were achieved: a physical mathematical model was designed, data processing and visualization programs were developed successfully and performance was validated with various experimental tests and trajectories, obtaining an ATE metric of 30.95 mm to 171.25 mm depending on the type of movement , showing the implementation of this project's proposal is effective for an indoor robot that doesn't need extremely precise but good measurements, and still remains beneficial in costs and independence from robot's kinematics.

In addition to what has been mentioned throughout the paper, future work will target the implementation of a focal correction introducing a lens to have a height independent separated from the floor system to avoid sensor damaging. To obtain a better adaptation to the robot currently under development, a change of micro-controller from Arduino MEGA to one with a smaller size could be considered, even if it involves a change of protocol and sensor model. It is also recommended to perform further analysis on the differences in the implementation of different mathematical models, sensor locations, evaluation metrics and ways of using redundancy or data fusion for three and more sensors to determine the best approximation.

5. Acknowledgements

We wish to thank professor Hugo Alberto Murillo Hoyos for his advice in the communication with sensors aspect, EAFIT University and their engineering technicians for lending the proper spaces and materials to develop the experimental tests, and finally, professor Elena Montilla Rosero for her advice in the way of arranging the specific objectives and presentation of the project.

References

[1] M. Dille, B. Grocholsky, S. Singh, Outdoor downward-facing optical flow odometry with commodity sensors (2010) 183–193.

[2] A. Paijens, L. Huang, A. Al-Jumaily, Implementation and calibration of an odometry system for mobilerobots, based on optical computer mouse sensors, Sensors and Actuators A: Physical 301 (2020).

[3] R. Ross, J. Devlin, S. Wang, Toward refocused optical mouse sensors for outdoor optical flow odometry, IEEE SENSORS JOURNAL 12 (2012).

[4] R. Li, C. He, Y. Tang, Improved modeling and fast in-field calibration of optical flow sensor for unmanned aerial vehicle position estimation, Measurement 225 (2024).

[5] A. Bonarini, M. Matteucci, M. Restelli, Automatic error detection and reduction for an odometric sensor based on two optical mice (2005) 1675–1680.

[6] A. Bonarini, M. Matteucci, M. Restelli, A kinematic-independent dead-reckoning sensor for indoor mobile robotics 4 (2004) 3750–3755 vol.4.

[7] D. Sekimori, F. Miyazaki, Self-localization for indoor mobile robots based on optical mouse sensor values and simple global camera information (2005) 605–610.

[8] M. Cimino, P. R. Pagilla, Optimal location of mouse sensors on mobile robots for position sensing, Automatica 47 (2011) 2267–2272.

[9] J. Palacin, I. Valgañon, R. Pernia, The optical mouse for indoor mobile robot odometry measurement, Sensors and Actuators A: Physical 126 (2006) 141–147.

[10] V. Sukhthanker, Vineetsukhthanker/fct3065-xy_mousesensor: Interface fct3065-xy optical mouse sensor with arduino, https://github.com/VineetSukhthanker/FCT3065-XY_MouseSensor, 2020.

[11] L. Patiño-LagaUni, Pose sensor with n optical mice, https://github.com/LagaUni/Pose-Sensor-with-N-Optical-Mice/blob/main/README.md, 2024.