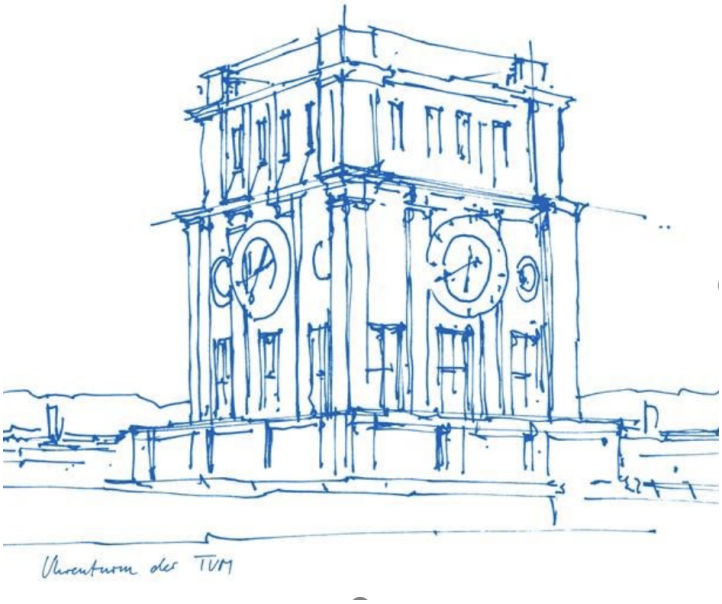
Gesture Control Drone

Tobias Coptil, Ngo Yan Abbie Fung, Mohamed Ali Gaoudi, Mohamed Amine Jradi, Louisa Siebel, Sindi Teqja

Project Report: Embedded Systems, Cyber-Physical Systems and Robotics
Bachelor of Information Engineering
Technical University of Munich



Professor
Amr Alanwar

Submitted on
20.08.2024

Contents

List of Figures

Abstract—

This project, developed for the Embedded Systems, Cyber-Physical Systems, and Robotics course at the Technical University of Munich, focuses on creating a gesture-based controller for a drone. The primary objective was to enhance and build upon a sensor's capabilities to accurately detect and interpret various hand movements. By integrating this sensor data with the drone's existing tracking and movement framework, we successfully established a seamless communication system that allows the drone to be controlled through intuitive gestures.

1. Problem Statement

Most drone controllers today come in the form of remote control devices or joysticks. These can be difficult to operate and are often unintuitive, which in turn limits precision and the natural feel of drone navigation. Those without technical expertise may feel that the complexity of these controllers creates a barrier, hindering a more immersive and enjoyable drone experience. With this in mind, the challenge was to find a more intuitive, user-friendly alternative to the traditional controllers. Inspired by various sources [1] [2], this project aims to address this issue by developing a sensor-based controller that interprets hand movements and gestures, converting them into commands, to control the drone more intuitively.

2. Literature Review

The field of drone control has undergone significant advancements, with conventional remote controllers and joysticks often being perceived as non-intuitive, particularly for users lacking technical expertise. This has prompted the exploration of alternative control methods, including gesture-based control systems, which hold the promise of a more natural and user-friendly interaction with drones. The existing literature on gesture-based drone control emphasizes the significant advancements and challenges in this field. While sensor-based systems demonstrate robust performance in diverse conditions, their integration and calibration are imperative to ensure responsiveness and accuracy. The combination of IMUs, sensor fusion algorithms, and PID control forms the foundation of these systems, facilitating precise and stable drone navigation through intuitive gestures. The integration of wearable technology further enhances the practicality of these systems, making them more accessible to users. As technology continues to advance, the potential for gesture-based control in drones and other robotics applications is vast, opening the door for more intuitive and user-friendly interfaces.

Methodology

2.1. Pre-Build considerations

Before our team could start building the controller for our drone, it was important to discuss the various steps and considerations we would have to take into account.

2.1.1. Sensor Fusion. Sensor fusion involves utilizing information from multiple different sensors in order to determine the state of a dynamic system ([3]). For the specific use case of the drone controller, this meant using Sensor Fusion 9-axis Inertial Measurement Unit (IMU) which fuses data from an accelerometer (measuring linear acceleration), a gyroscope (measuring rotational velocity), and a magnetometer (measuring orientation relative to the Earth's magnetic field). By taking these individually and fusing their data, a far more precise and stable estimation of the drones orientation could be achieved. This data fusion requires using algorithms such as the Madgwick or Mahony filter but different algorithms had to be tested in order to find the best possible fit for this project.

2.1.2. PID Control. When working with a drone or any other vessel that requires stabilization, it is always important to think about PID control. PID control is a continuous mechanism that uses Proportional, Integral, and Derivative (PID) feedback to maintain the desired output of a system ([4]). A PID controller takes the fused data from the sensor fusion IMU and uses it to ensure stabilization by calculating any necessary adjustments the drone may have to make, due to external forces such as wind. The controller would adjust the drone's motors to counteract this and bring

it back to its desired orientation. However since the drone that was used is a Tello drone with built-in stabilization and control algorithms, building a PID controller from scratch was not necessary. That being said, this is an important aspect to take into account when building a controller.

2.1.3. Controller Hardware. It was decided that the controller itself would consist of an Arduino Nano 33 BLE Sense ([5]) sensor and a battery to allow for freehand use of the sensor as the Arduino requires a near constant power supply. A 3D printed case would be used to house the battery and sensor together and a glove to allow the controller to be wearable and portable. This design idea allowed for an easy yet effective controller to be constructed.

2.2. Building and Testing the Controller

2.2.1. Sensor Calibration. At the core of this project is the sensor which serves as the controller for the drone. It is thus integral that the sensor knows the drones position at all times. Therefore, the first step in building the controller was sensor calibration. This process required finding the best solution for the specific use case of the project and took the evolution detailed below.

Gyroscope (initial issues)

Initially we experimented with using only a gyroscope in order to see how far we would be able to get with this alone. We also wanted to observe how accurate the gyroscope is. However, we encountered some problems not long after we started experimenting. Firstly, we noticed that the sensor was unable to immediately determine its initial orientation. The gyroscope required a second or two to "settle" and stabilise its readings. This meant that doing two consecutive movements would result in the new position being calculated based on the previous one and not its actual initial position, which isn't accurate. To fix this, we adjusted the calculation method. We added an offset so there is always a starting point it starts from and comes back to. We also added ranges because it is very difficult to set the neutral state only at 0 when the hand movement is a sensitive gesture. Instead each range of values of rotation corresponded to a state (neutral, right, certain speed...). This helped improve the accuracy of its readings but still did not solve the problem altogether. In addition to the aforementioned problem, we also observed that the sensor started to "drift.", meaning that even if the sensor was held still, it would slowly begin to behave as if it was moving, causing the readings to become inaccurate. This occurs as the gyroscope accumulates small errors over time and as these errors build up we get the 'drift'. We found that a possible solution to this was adding additional sensors to help reset and correct the orientation, keeping the direction readings accurate over time.

Madgwick observation

In order to address the 'drift' issue, we decided to combine the gyroscope with an accelerometer and also a magnetometer as we thought an accelerometer addition alone was not enough, due to the fact that it is unable to determine any reference direction, which we then defined as the North using the magnetometer. The decision to use multiple sensors meant that we now had to perform sensor fusion. There are a few algorithms to do this operation but after some research we decided that the Madgwick algorithm [6] was the most suitable in our case. The Madgwick algorithm proved to be accurate but not exactly what we needed. We observed a delay in how the algorithm responded to hand movements, meaning that the drone did not react immediately when the hand tilted. The changes in the calculated orientation varied exponentially, which lead to small tilts of the hand causing rapidly increasing responses, making the drone harder to control accurately. We then tried to adjust the sensitivity in the hopes that this would solve the issue, however, the algorithm still did not behave as expected and the values just rose exponentially faster with no real regard to the actual degree of hand movement, leading to erratic control rather than the smooth, real-time response that we required. It was thus time to consider other alternatives.

Second Fusion Observation

After trying the Madgwick algorithm, we

explored a simpler approach by using only an accelerometer and a gyroscope. This method turned out to be much more effective for our use case. Unlike the Madgwick algorithm, this simpler fusion provided real-time, accurate responses to hand movements. After normalising the values and setting a range for them, the sensor readings changed immediately when the hand holding the sensor was tilted and it directly reflected the degree of rotation.
We used this formula to do the roll and pitch:

$$\text{roll} = \arctan\left(\frac{y}{z}\right) \times \frac{180.0}{\pi}$$

$$\text{pitch} = \arctan\left(\frac{-x}{\sqrt{y^2 + z^2}}\right) \times \frac{180.0}{\pi}$$

This one for the vertical by setting a baseline (an offset):

$$\Delta y = y - \text{baseline}_y$$

We also set the yaw and included a sensitivity parameter according to our need. We fixed it through testing:

$$\text{yaw} += z \times \text{sensitivity}$$

2.2.2. Bluetooth Connection. Getting the Arduino sensor to take the correct readings was only the first step in using it as a controller. The next step in the project was sending this data to a computer in order for us to be able to use it to control the drone. This Arduino-computer connection was set up using a library called BLEAK (Bluetooth Low Energy platform Agnostic Klient). This library is "capable of connecting to BLE devices acting as GATT servers." [7]. We had some trouble finding the MAC address of the Arduino sensor as this was required for the code but we used the nRF Connect app [8]in order to test if the Arduino was in fact advertising and also to obtain its MAC address.

2.2.3. Signal Translation. After setting up the connection between the computer and the Arduino sensor, we now needed to ensure that the connection between the computer and the Tello drone is stable. The Tello Drone comes with a Wi-Fi connection function which allows the user to connect the drone to a laptop with very little difficulty [9]. Once the computer has received the IMU sensor data from the Arduino Nano 33 BLE Sense to calculate roll, pitch, yaw, and vertical movement. It then sends this data over BLE, to a connected central device which in this case is the PC. The sensor continuously updates its orientation data and transmits it while the laptop is connected. The Arduino sends the coordinates serial prints to the laptop as a string which is received by the python script. It then receives the data as 4 values that will be parsed and then mapped to the drone command range (-100,100). This is then passed to the method contained in the djitello library (send_command()) which sends the commands to the drone (rc_commands). In this way, the drone is able to be controlled from the sensor.

2.2.4. Gesture Recognition and Integration. Having the connection between all of the components set up, concludes the basis work of the project. The next goal for this project was to incorporate gesture recognition. This means that the drone would respond to specific 'gestures' that the user made with their hands, therefore, we had to train a machine learning model to recognise gestures. It was decided to set a flex gesture as the take off command and a certain vertical rotation of the hand as the flip command. By performing the gesture 100s of times, we established our data.This can be seen in the accelerometer and gyroscope graphs for flip and flex in Figures: 2, 3, 4, 5.

In order to establish the data we performed the gesture and let the Arduino record the first 119 frames of said gesture. Recording of the frames would only start at a certain acceleration threshold so it only records the movement that we performed. The purpose is to perform an intense movement that the Arduino can detect without confusion. We constructed the data into a csv file composed of 6 main values which represent values of

each gesture on the 3 axes recorded by the gyroscope and the accelerometer.

After recording the data, we trained the data. We 60 of the data as training data, 20 for evaluation and 20 for testing. After trying this process multiple times to see which was the best recording of the data and the best number of iterations to train our neural network, we had a very satisfying result.

We had 100% F1 score, as shown by Figure 6 and Figure 1. All we had to do was convert this data to a header file that the Arduino could understand ([10]). We converted the model to an .h file. Using the ArduinoTensorFlow library we were able to read this file and work with the model.

Figures 8 and 9 overall, show a very successful training process, but the near-zero loss warrants some additional verification to ensure the results are valid and meaningful for the specific problem being addressed.

By running the final test (performing the gesture and gauging the ability to perform it) on out model, we found that the gestures were 100% detected accurately [11]. We did have an issue with converting the model to header file so we used an older version of tensor flow in googlecollab and extracted the .h file from there.

Combining this with the drone command, the Arduino detects a flex or flip and it sends a serial print of a "flex" or "flip" to the drone. For the take off, after establishing a successful bluetooth connection with the Arduino the laptop listens for a certain time frame till it detects a flex serial print. If nothing is sent it will attempt to listen 3 times before it throws an error and exits. Once the flex is detected the drone takes off through the take off command sent by the laptop.

As for the flip command, the approach is still not perfect as the drone may destabilize while attempting a flip as it confuses it for a normal hand movement to control the drone. Essentially what we did is set an acceleration threshold so the Arduino knows that this is a gesture to recognise not to a value to be transmitted. After the gesture is recognised, a serial print of "flip" will be sent. Once the laptop receives it, it send a flip command to the drone.

2.2.5. Case construction. The last section of this project consisted of making the case and glove that would carry the sensor and its battery. The glove we used was just a simple compression glove. After taking some time to design a box to house the Arduino and the battery, we decided that the Arduino should have its own case in order to ensure that the pins on the sensor remain protected. We found a ready made design for the Arduino case and had it 3D printed [12]. After some trial and error with 3D printing, which involved changing measurements and some minor adjustments, we finally had a finished design, as shown in Figure 7. We decided to position the Arduino on top of the glove, while the battery is positioned at the bottom, essentially in the palm of your hand. This also allows space for the wires which connect the battery to the sensor. This set up can be seen in Figure 10 and Figure 11

3. Results

Through the successful development of a gesture based drone controller in this project, the usability and intuitiveness of drone navigation have been greatly improved. We were able to create a precise and responsive control system by combining several sensors and using sensor fusion techniques.

Extensive testing and calibration were conducted during the development process to guarantee accuracy and stability, tackling frequent problems such sensor drift and delayed response. We have successfully implemented a system where flexing the hand allows the drone to take off and then be controlled. The rotation of the hand governs the drone's directional movement, while vertical hand movements control the drone's altitude.

That being said, we are currently facing a challenge with the flip gesture. The Arduino successfully recognizes the flip gesture, but we are still working on preventing it from disrupting the drone's movement. Because the data is transmitted in real-time, the drone often destabilizes before the flip gesture is fully

detected. Additionally, there are some redundant values being transmitted during gesture recognition, which can affect the accuracy and responsiveness of the control. To address these issues, we have researched a solution where we put in place an acceleration threshold that, when applied to particular maneuvers, pauses data transmission, enabling the systems to detect gestures without causing the drone to lose stability. This pause in data transmission allows the system to focus on accurately recognising the gesture before resuming control, thereby preventing destabilization and improving the overall reliability of the drone's movements.

4. Conclusion

The project showed that hand gestures may be used to control a drone in an efficient manner, increasing user accessibility and engagement. Drone technology can become more accessible to a wider audience by lowering entrance barriers for individuals who are not familiar with conventional remote controllers with this user-friendly control method. As a result, this innovation holds significant promise for applications in various fields, such as photography, surveillance, and recreational activities.

We faced and overcame a number of obstacles during the development process. The flip gesture interfering with the drone's stability being one of the only issues we could not fully solve. Additionally, the experiment offered insightful information about the fusion of various hardware and software components. The possibility of wearable and portable technology in managing intricate systems was demonstrated by the utilization of the Arduino Nano 33 BLE Sense in conjunction with Bluetooth Low Energy (BLE) connectivity. The controller's practicality and comfort were enhanced by the glove and case designs made by 3D printing our,using CAD software, self-made objects which guaranteed a seamless and efficient user experience.

In conclusion, this study opens the door for more user-friendly interfaces in future applications by showcasing the possibility for creative control techniques in robotics and embedded systems. Not only can the effective usage of a gesture-based drone controller improve user experience, but it also paves the way for further developments in the sector. The project's proposed ideas and solutions have broad applicability, which will spur additional creativity and accessibility in the field of human-machine interaction as technology advances.

Appendix

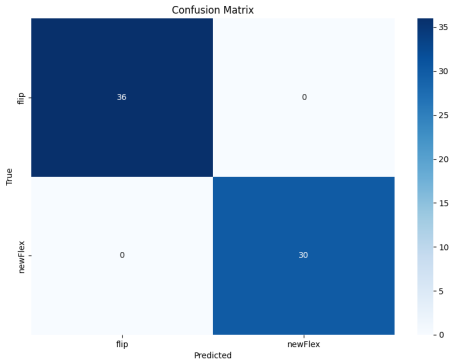


Figure 1: Confusion Matrix

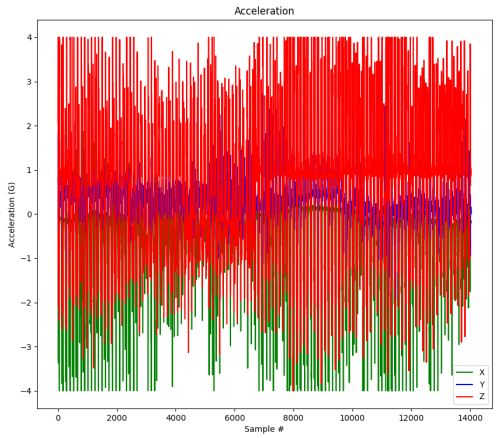


Figure 2: Flex Acceleration

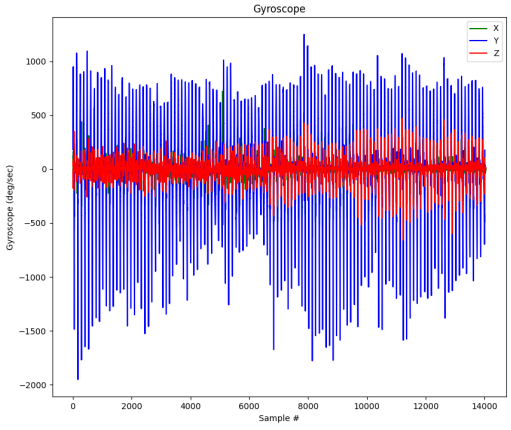


Figure 3: Flex Gyroscope

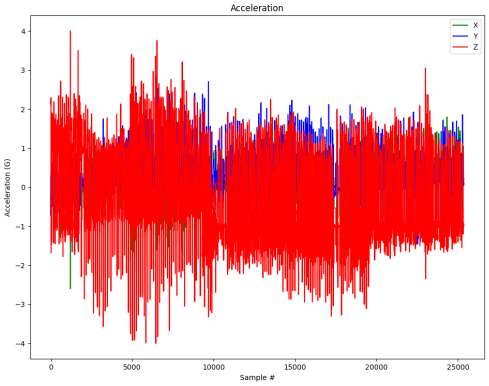


Figure 4: Flip Acceleration

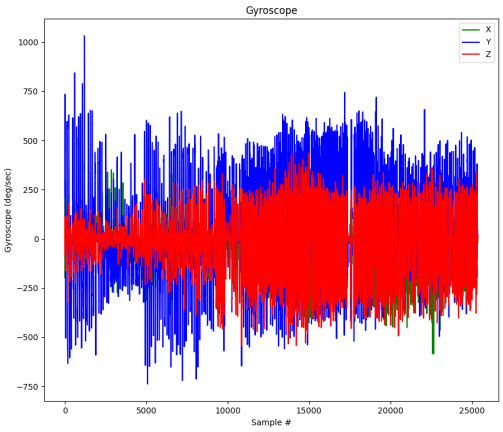


Figure 5: Flip Gyroscope

```
  F1 Score: 1.00
newFlex:
  Precision: 1.00
  Recall: 1.00
  F1 Score: 1.00

Classification Report:
              precision    recall  f1-score   support

        flip       1.00      1.00      1.00        36
     newFlex       1.00      1.00      1.00        30

    accuracy                           1.00        66
   macro avg       1.00      1.00      1.00        66
weighted avg       1.00      1.00      1.00        66
```

Figure 6: F1 Score

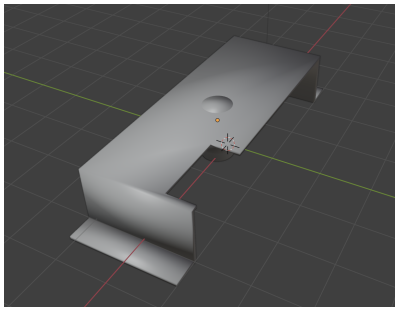


Figure 7: 3DModel

Figure 10: Glove Design: Bottom view

Figure 11: Glove Design: Side view

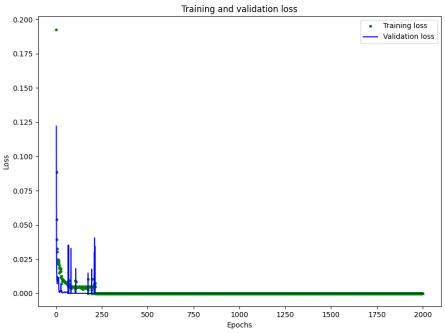


Figure 8: Training And Validation Loss

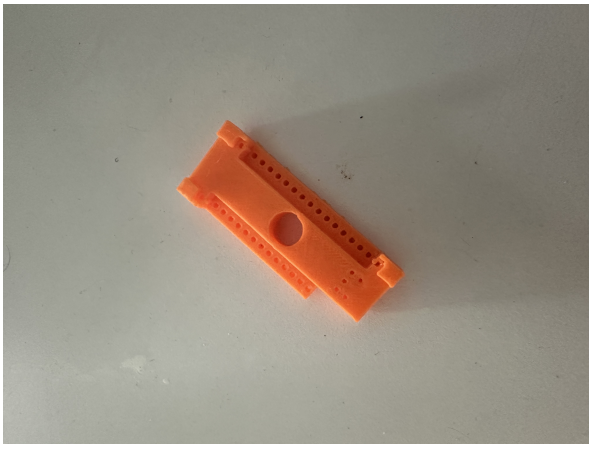


Figure 12: Case to hold the Arduino

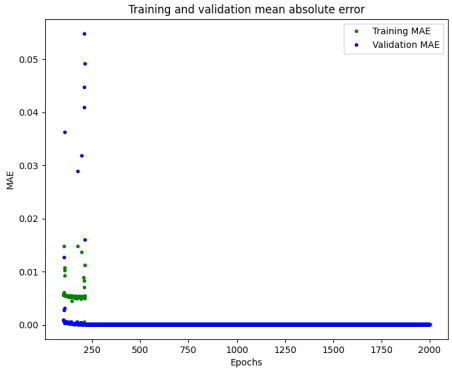


Figure 9: Training And Validation Mean Absolute Error

References

[1] Bitluni, "Gesture controlled drone: A beginner's guide," YouTube, 2023, accessed: 2024-05-06. [Online]. Available: https://www.youtube.com/watch?v=zPKZQx0ZVxY

[2] S. Chakraborty, "Gesture controlled drone (part 1)," Hackster.io, 2022, accessed: 2024-05-06. [Online]. Available: https://www.hackster.io/csoham/gesture-controlled-drone-part-1-bdf9d1

[3] ScienceDirect, "Sensor Fusion," ScienceDirect, 2024, accessed: 2024-05-11. [Online]. Available: https://www.sciencedirect.com/topics/engineering/sensor-fusion

[4] National Instruments, "PID Theory Explained," https://www.ni.com/en/shop/labview/pid-theory-explained.html, 2024, accessed: 2024-07-03.

[5] Arduino, "Arduino Nano 33 BLE Sense with Headers," https://store.arduino.cc/products/arduino-nano-33-ble-sense-with-headers, 2024, accessed: 2024-05-06.

[6] Read the Docs, "Madgwick filter," Read the Docs, 2024, accessed: 2024-05-11. [Online]. Available: https://ahrs.readthedocs.io/en/latest/filters/madgwick.html

[7] H. Blidh, "Bleak: Bluetooth Low Energy platform for Python," https://github.com/hbldh/bleak/tree/develop, 2024, accessed: 2024-06-17.

[8] Nordic Semiconductor, "Nordic semiconductor documentation," Nordic Semiconductor, 2024, accessed: 2024-05-17. [Online]. Available: https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/index.html

[9] Ryze Robotics, "Tello SDK 2.0 User Guide," https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf, 2024, accessed: 2024-05-07.

[10] Anonymous, "Google Colab Notebook," https://colab.research.google.com/drive/1Jj_LjUFtrzhLT6QaVmfLupGtPSITQi5Q, 2024, accessed: 2024-07-10.

[11] D. Gaoudi, "Drone gesture control - gesture recognition," GitHub, 2024, accessed: 2024-06-17. [Online]. Available: https://github.com/DaliGaoudi/DroneGestureControl/tree/main/gestureRecognition

[12] P. U. Domino2, "Arduino Nano 33 Test Adapter," https://www.printables.com/en/model/600759-arduino-nano33-testadapter, 2024, accessed: 2024-07-11.