IMPERIAL

Wireless Gesture-based Controller for Drones

Interrim Report

Author

Nik Lewis

CID: 02031260

Supervised by

Dr Ed Stott

Second Marker

Dr TBC

An Interim Report submitted in fulfillment of requirements for the degree of
Master of Engineering in Electronic and Information Engineering

Department of Electrical and Electronic Engineering
Imperial College London

2024

Abstract

Increasing complexity of Unmanned Aerial Vehicle (UAV)s is, in many cases, leading to steeper learning curves for controlling them. In many cases, operators have to undergo special training to learn how to control the UAV, and it takes many hours to be able to do so with precision. Furthermore, operation of a drone often requires 100% of the user's attention, rendering them unable to do anything else during operation.

This project aims to create a new form of drone controller, sacrificing a degree of precision and versatility for increased freedom for the user. The proposed design is one that can be worn on the back of the hand, freeing up the user to engage in other activities as the controller passively detects certain gestures and interprets them as controls. The project also aims to make the design inclusive, in an attempt to allow users who, for whatever reason, may not be able to use traditional drone controllers.

Declaration of Originality

I hereby declare that the work presented in this thesis is my own unless otherwise stated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced.

Copyright Declaration

Contents

List of Acronyms

UAV Unmanned Aerial Vehicle

the Supervisor Dr Ed Stott

HH Hand Held

AR Augmented Reality

HMD Head-Mounted Device

HRI Human-Robot Interaction

UI User Interface

IMU Inertial Measurement Unit

RMSE Root Mean Squared Error

EMG Electromyography

SVM Support Vector Machine

KNN K-Nearest Neighbour

DoF Degrees of Freedom

PID Proportional-Integral-Derivative

ML Machine Learning

HTTP Hyper Text Transfer Protocol

SDK Software Development Kit

UDP User Datagram Protocol

TCP Transmission Control Protocol

API Application Programming Interface

MPU Motion Processing Unit

SPI Serial Peripheral Interface

I^{2}C Inter-Intergrated Circuit

DMP Digital Motion Processor

FIFO First-In-First-Out

IoT Internet of Things

ESTOP Emergency Stop

C2 Server Command and Control Server

CLI Command Line Interface

HMM Hidden Markov Model

TLX Task Load Index

List of Figures

1 Introduction

Contents

1.1 Overview of Progress

This report summarises the progress made during the first term of the project. The majority of construction for the first hardware prototype has been completed, and data is now being extracted from the controller. As with any Machine Learning (ML), before doing detailed research into various algorithms and approaches, it is prudent to first examine the data that you are working with. Therefore, the primary objective of the first term has been implementing basic hardware to extract data from the controller. This may not seem a typical approach to a final-year thesis but was deemed appropriate throughout regular meetings with Dr Ed Stott (the Supervisor).

1.2 Interrim Report Breakdown

This project follows an iterative process - research leads to experiments which lead to conclusions and further research. This gives rise to the question of what to include in the traditional ”background” section of the report, since much of the research conducted will be a result of design

decisions made in the implementation. In an attempt to avoid a fragmented narrative in the report, the structure is as follows:

- **Background**: A summary of the relevant background reading completed to date, necessary for a high-level understanding of the controller's functionality. This includes information on any components and concepts that were considered at some point during the implementation process. Design decisions are not justified in this section, but a higher level of detail is provided for sections that ended up being more relevant during implementation.
- **Specification**: A comprehensive list of both the requirements and desireables, agreed upon in biweekly supervisor meetings during the first term of the 2024-2025 academic year.
- **Project Management**: A summary of the project management decisions made (to date), including a projected time schedule and a project-level risk assessment. Ethical, legal and safety considerations are also outlined in this section.
- **Implementation**: Documentation of the progress made with implementation (to date).
- **Evaluation Plan**: This section also contains considerations for how the project should be evaluated throughout implementation, moving forward.

2 Background

Contents

2.1 DJI Tello Drone

The drone that the project will aim to control will be the DJI Tello. The *Tello*, henceforth referred to as the ”drone” for simplicity, is an educational quadcopter UAV developed by DJI and Ryze that can be interfaced with over Hyper Text Transfer Protocol (HTTP) sockets. The drone itself will not be a large focus of the project, as the aim is to create a gesture-based controller that could, in theory, be modified to accommodate any other drone communication protocol.

Figure 2.1: *Tello* drone as sold by DJI [1]

2.1.1 SDK

DJI provide a public python Software Development Kit (SDK) [2], [3] for interfacing with the drone. This includes examples contained within a public GitHub repository [4] that is, for the most part, accurate and simple to use. This background material will not go into detail about how the SDK works, it is sufficient to know that controls are streamed over User Datagram Protocol (UDP) packets that are generated using the provided Python Application Programming Interface (API). The interested reader may consult the user guides [2]–[4].

2.1.2 Tello Control App

As an educational drone, the DJI Tello does not ship with a dedicated controller, but instead relies on the user installing an app and using that as a Hand Held (HH) controller (see figure 2.2). This provides two virtual joysticks - one for elevation/yaw and one for pitch/roll. A separate button is used for takeoff/landing. Controls regarding photography and videography can be seen in the top-right corner. As expected, this form of controller requires a high level of concentration in order to control the drone - particularly if using on a device with a small display.

The app also shows diagnostics at the top of the screen - this is critical in order to be able to safely operate the drone. It includes the following:

- Battery indicator
- Wi-Fi indicator
- Bluetooth status
- Drone odometry data including velocity and elevation

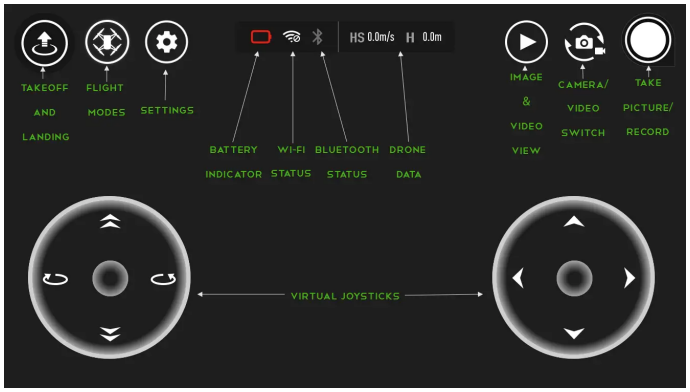

Figure 2.2: *Tello* control app as annotated by [5]

2.1.3 DJI Mavic Hand-Held Controller

On the other end of the consumer spectrum, there are robust portable HH controllers for drones such as the DJI Mavic 3 Pro [6]. These are of a higher build quality, and come with the bespoke *OcuSync* [7] DJI communication protocol. This has considerable range advantages over Wi-Fi (reaching ranges up to 20km), though recent work [8] has shown that possible security vulnerabilities do exist. The main focus of this project will not be security, however, the interested reader may consult [8], wherein the authors expose how information about a drone's location can be reverse-engineered from certain aspects of the OcuSync protocol.

Figure 2.3: *DJI RC Pro* controller as per [9]

2.2 Existing Research into Alternative Robotics Controllers

2.2.1 Augmented Reality Controllers

Within the robotics research space, there has been recent investigation into alternative methods of robot control. One such avenue of research is the use of Augmented Reality (AR) Head-Mounted Device (HMD)s [10], [11]. While [10], [11] aim to improve the User Interface (UI) for controlling *"legged manipulators"*, the principles and challenges of Human-Robot Interaction (HRI) remain largely identical.

Study [10] explores in great detail multiple HRI metrics, including the *"cognitive workload"*, *"system usability"* and *"technological acceptance"*. The authors developed an AR-based UI for controlling a *Boston Dynamics Spot* robot, and conducted a detailed study with 17 participants. Results showed that while traditional HH controllers performed better in terms of time and cognitive workload, the AR UI significantly enhanced immersion and user engagement.

Following on from this, study [11] extends the sample size to 27 participants, and the authors carry out a more robust *"Bayesian data analysis"* across key dimensions: cognitive workload, technology acceptance, fluency, system usability, immersion, and trust. Upon observing users' natural tendencies when operating the robot, the author where able to incorporate various *"cognitive offloading"* techniques which lead to equal and occasionally even better performance than HH controllers in navigation tasks.

These papers advance the understanding of AR UIs for robot control, highlighting their potential to enhance immersion and usability when optimised for cognitive offloading. However, for now, traditional HH controllers retain advantages in simplicity and familiarity.

2.2.2 Controllers with IMUs

The use of Inertial Measurement Unit (IMU)s as a potential replacement for joysticks is not a new concept [12]–[16]. Sensor fusion algorithms have advanced enough to be able to create stable 6-Degrees of Freedom (DoF) pose estimation applicable to a variety of robots. In [12], a 6-DoF IMU was evaluated as part of a microsurgery controller, and found significant improvements in the Root Mean Squared Error (RMSE) during trajectory-following tasks.

The authors of [14] introduce a wearable IMU-based teleoperation system designed for controlling collaborative industrial robots through body movements. The system utilises a network of IMUs placed around the upper body and torso. Through statistical methods comprised of various correlations and transforms, human gestures were accurately mapped to robot commands. This provided a means of controlling a robotic arm that was flexible and user-friendly, requiring much less training than traditional methods.

Shin et al. [13] explore gesture recognition using an IMU and Electromyography (EMG) sensor. Euler angles were converted to 2D vectors represented by magnitude and argument, and this data was fed into a trained Hidden Markov Model (HMM) for gesture recognition. The purpose of the EMG sensor was to denote the beginning and end of a "gesture" - the device would only attempt to recognise gestures when the user's hand was closed.

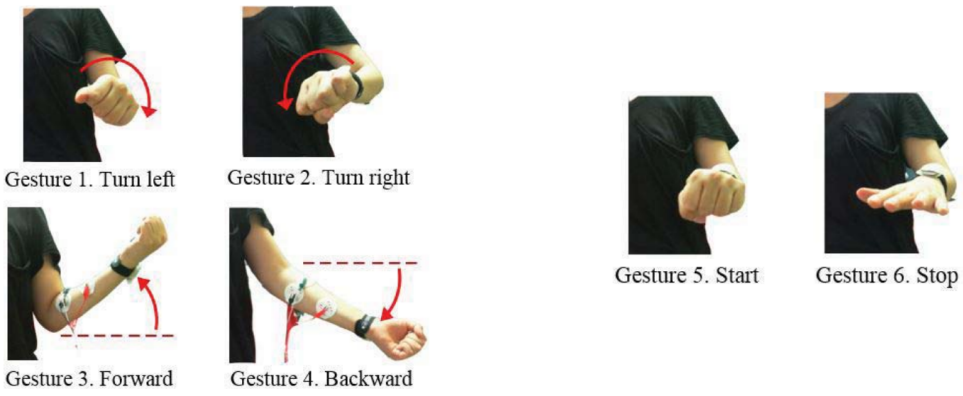

Figure 2.4: Gestures used by Shin et al. [13]

The authors demonstrated apparently successful gesture recognition. Use of the EMG sensor to clearly indicate the start and end of a gesture proved to be very convenient. However, only the *pitch* and *roll* were used, limiting the action space to 2 dimensions. Extending the system to react to yaw would greatly increase the range of possible gestures.

The paper [13] remarks that limited accuracy was achieved using a single stage HMM (see section on Hidden Markov Models for more detail), and that extending this to 2 stages would be a valuable improvement. The authors heavily advocate HMMs as a means of gesture recognition, but there is limited evidence that the device was successful for multiple different users.

A more advanced controller has been presented by [15], further exploring gesture recognition using ML. The controller implements two control modes: *direct mode*, in which users can control the drone using the pitch, roll and yaw of the IMU, and *gesture mode*, in which pre-defined gestures

mimic helicopter guidance signals (figure 2.5) for easier control. Several methods of classification are analysed, including Ensemble, Support Vector Machine (SVM)s, K-Nearest Neighbour (KNN) and Naive Bayes.

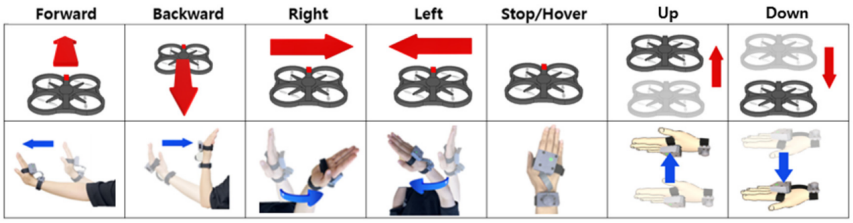

Figure 2.5: Gestures use in [15], inspired by helicopter hand signals

Upon comparison of the classification methods, the authors found that the *Ensemble* method achieves the highest accuracy of 97.9%. Unlike [13], Lee and Yu extract spectral peaks from the IMU data as an additional input for the ML models. This brings the ML concept of feature extraction to the table, which previous studies lacked. The system also integrates *"vibrotactile feedback"*, alerting users of obstacles nearby using haptics. Both simulation and real-world experiments were preformed with multiple participants, and the authors concluded the controller to be an intuitive and effective interface for drone operation.

It should be noted while [13] uses the EBIMU24G (a wireless module with an IMU integrated) to obtain odometry data, [15] uses a simple SEN-14001 SparkFun breakout module with great success. This demonstrates that gesture recognition capability is possible using low-cost IMUs. In both studies, the sensors have 9 DoF, seeing as the presence of a magnetometer can greatly improve sensor fusion capabilities (see section on Sensor Fusion). One curious aspect that does not seem to be mentioned is whether or not the vibration affected the measurements on the IMU.

2.3 Hardware

In order to recognise gestures made by the end-user, it is necessary to collect some sort of data from the controller in order to observe its "state". This needs to depict the position and motion of the user's hand with sufficient accuracy and dimensionality so that a ML model can be trained with it. At the same time, this should be achieved with as few sensors as possible. There is a clear tradeoff in this respect: The controller should remain easy to use and not obstruct the user from using their hand for other tasks.

2.3.1 MPU-9250

The *MPU-9250* and *MPU-6050* are two low-cost motion sensor modules that are often used in prototyping wearable devices [17]. Both modules contain an accelerometer to measure linear acceleration, and a gyroscope to measure angular velocity. In addition to this, the MPU-9250 also contains a magnetometer, though it can be easily used without it. A Motion Processing Unit (MPU) can be interfaced with using either the Serial Peripheral Interface (SPI) or Inter-Intergrated Circuit (I^2C) protocol, making it highly versatile for use with micro-controllers. Both MPUs come with a DMP unit onboard, which calculates quaternions using a proprietary sensor fusion algorithm. Inspection of the register map [18] reveals that the DMP populates a First-In-First-Out (FIFO) register with this data, which can then be polled using I^2C.

2.3.2 Flex Sensors

Flex sensors are variable resistors that change their resistance when they are bent. These are commonly used in applications in glove controllers, particularly on the back of the fingers. The sensors used in this paper are the *Adafruit Short Flex Sensor* [19].

2.3.3 Particle Photon 2

The Particle Photon 2 [20] is an Internet of Things (IoT) capable micro-controller that is used in the early development process of the controller. The model was chosen since it was already available at the time. Most *Arduino* libraries are compatible with it and you can easily upload `.ino` and `.cpp` files. The Photon 2 can be powered using a 3.6-4.2V DC power supply [20], or a USB power supply.

2.4 Sensor Fusion

An IMU typically contains 3 sensors: an accelerometer, a gyroscope, and a magnetometer (optional). Each of these has their limitations: accelerometers are sensitive to noise and external forces in the environment, gyroscopes drift over time due to integration errors, resulting in an accumulated bias. Magnetometers are prone to interference from nearby devices, and even the earth's magnetic field.

Sensor fusion aims to solve these problems by combining the data of each sensor in a way that mitigates these effects, and allows the extraction of the 3 fundamental Euler angles: *pitch*, *roll* and *yaw*. This section will briefly outline some common techniques used with IMUs such as the *MPU-9250*.

2.4.1 Complementary Filter

The basic concept of a complementary filter is to design a transfer function for each sensor that filters out unreliable frequencies, and then combine the results of each sensor together. In the context of IMUs, a gyroscope is generally reliable at high frequencies, and an accelerometer is reliable at low frequencies [21]. In order to get a better estimation of the current attitude (`angle`) of a device, we can use the following:

`angle = (1 - `α`) * (angle + gyroscope * dt) + `α` * accelerometer`

Where α can be thought of as a 'trust' factor, i.e. how much we value the reading from each sensor. A common value for α is 0.98. The complementary filter is often used in simple applications for low-noise environments, but will certainly not suffice given the complexity of gesture recognition. For this reason it will not be considered further, but the interested reader may refer to literature such as [21] for a more in-depth explanation of the complementary filter in the frequency domain.

2.4.2 Kalman Filter

A Kalman filter takes the next step by considering the evolution of the system over time (see *intuitive approach to the KF*, [22]), in order to minimise the error covariance matrix. A Kalman filter is split into two stages: First, the system predicts the next state. Then, once the output of the system is known, the previous prediction is corrected as required. These are commonly referred to as "prediction" and "update" [23]. The correction is made by adding a *weighted difference* between the predicted and actual state. Requiring many tunable parameters, Kalman filters are a broad topic that could easily occupy the entirety of a final year project, so the background reading was largely focused on comparing it to other techniques [23].

2.4.3 Madgwick & Mahony Filters

Both the Madgwick and Mahony filter are popular IMU orientation-estimation algorithms. Both filters use *quaternions* to represent orientation, and employ various techniques for fusing gyroscope and accelerometer data [23].

The Mahony filter uses the concept of Proportional-Integral-Derivative (PID) control to correct IMU measurements. As such, the parameters K_p, K_i and K_d representing the proportional, integral and derivative gains respectively, all require tuning. Mahony filters are widely known for impressive disturbance rejection in environments with vibration or other interference. For this reason, they are frequently employed in the robotics and aerospace industries.

The Madgwick filter calculates the error between a predicted quaternion and the actual quaternion, then uses that error to update the next estimate. This is similar to a Kalman filter, though a Madgwick filter uses *gradient descent* to update the estimate, and relies on a single tuning parameter β to balance noise suppression and performance.

2.4.4 A Comparison

The authors of study [23] performed experiments using all the aforementioned sensor fusion techniques. The study evaluates their accuracy, stability, and computational efficiency in both static and dynamic conditions. Results show that the Kalman filter offers the best accuracy, while the Mahony filter is the most computationally efficient (roughly 30% faster), while still providing comparable results.

2.4.5 Using the DMP

As previously mentioned in the previous section, the MPU-9250 contains a Digital Motion Processor that calculates filtered quaternions on-chip and writes them to a FIFO that can be polled. Whilst this 'out-the-box' solution may be good enough for many scenarios, there is no way to tweak the parameters of the filter - nor is the algorithm even publicly known.

2.5 Hidden Markov Models

A HMM is a statistical model that updates *hidden states* based on observable outputs. Below are the fundamental concepts:

- **States** - a state represents a unique condition of the system, just like in a state machine.
- **Observations** - Observable data linked to hidden states.
- **Transition Probabilities** - The likelihoods of moving between states, specific to each individual state
- **Emission Probabilities** - The probabilities of producing an observation given a state.
- **Initial State Probabilities** - The starting likelihood of each state, upon system initialisation.

A *hidden stage* can be thought of as an internal classification layer. HMMs can have multiple hidden stages, and there is often a tradeoff here: single-stage HMMs may struggle with complex patterns due to lack of contextual information. However, multi-stage HMMs become increasingly computationally demanding due to more sequential processing. There is evidence of both single [13] and two-stage [16] HMMs being used for gesture recognition. While Shin et al. recommended a two-stage HMM as future work, it should be noted that the referenced paper [16] was built around a system of much higher complexity - employing a *Microsoft Kinetic 3D Camera* to detect gestures.

In the latter, the recognition process was broken down into two stages: first identifying prime gestures, then classifying whole tasks. The first stage acts almost as a level of feature classification, decomposing a movement into a sequence of sub-movements. Using this approach, the system achieved an accuracy of 95.33%, significantly higher than a single-stage HMM used in the same application.

3 Specification

Contents

The specification points are spilt into those that concern the controller, those that concern the server, and those that concern the system as a whole. *Requirements* are prefaced with the letter **R**, these are concrete specification points that need to be achieved in order for the project to be considered successful. *Desireables* are prefaced with the letter **D** and indicate parts of the specification that should ideally be achieved, but might not due to time and complexity constraints.

3.1 The Controller

R1. The controller should be unintrusive to regular hand use. This means the user should still be able to pick up objects and use everyday items (for example, a smartphone) without having to remove the controller.

R2. Gesture recognition must be passive, and not dependent on the user indicating the start of a motion using a hard input

R3. The controller must have a manual mode that can control the drone's pitch, roll, yaw, and elevation through tilting of the hand.

R4. The controller must be able to recognise a minimum of 5 actions - 2 for takeoff and landing, and 3 that can be mapped to other custom drone commands. These will be referred to as the

Takeoff gesture, Landing gesture, and programmable gestures 1-3 henceforth.

R5. The controller must have an Emergency Stop (ESTOP) button on the back of the hand. This needs to trigger a failure state that is well-defined and lands the drone as safely and quickly as possible.

R6. The controller must be able to control the drone through a middleman C2 Server (i.e. a laptop) for additional safety at the cost of reduced versatility.

R7. The controller should have the ability to be battery-powered

R8. The controller should be robust in design and look like a marketable product

D1. The controller might be able to run stand-alone, without the use of a middleman server. It would interface directly with the drone over websockets.

3.2 The C2 Server

R9. The server must display health metrics and the current state of both the drone and the glove. ”Health metrics” must include at minimum the battery of the drone, and connection status of both the drone and controller.

R10. The server must be able to manually take control of the drone and remote-override the controller

3.3 The System

R11. The failure state in which the glove has disconnected from the server should be well-defined and handled by the server

R12. The failure state in which the drone has disconnected from the C2 Server should be well-defined and the user should be warned appropriately (there is little that can be done if the drone has disconnected)

R13. Gesture recognition must work with a variety of users. This may be done through means of an initial calibration for each new user profile.

D2. The end-to-end latency between gesture initialisation and drone reaction should be as low as possible

4 Project Management

Contents

Due to the agile nature of the project, the timeline of the project is done at a relatively high-level. Following a common agile methodology used in industry today [24], [25], the project is split into several "*Epics*", each representing a separate aspect of the system. Each *epic* is split into several "*Story*" tasks. Finally, a *story* will be split into sub-tasks which will be concrete, well-defined and well-scoped tasks that can be accomplished within a small timeframe.

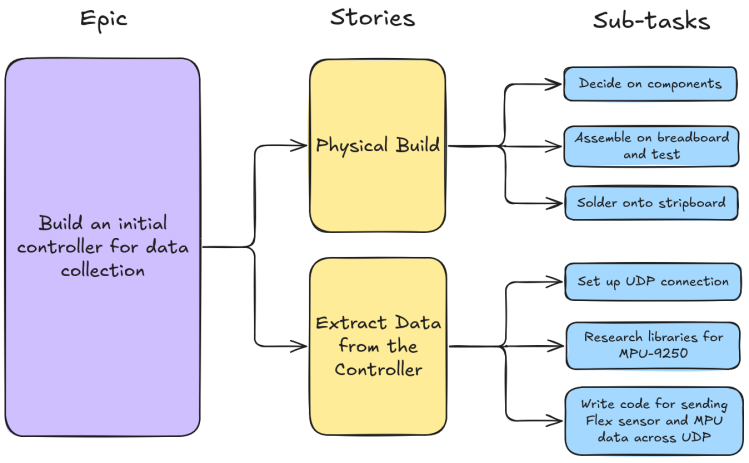

Figure 4.1: An example of how a high-level *Epic* task can be decomposed into *stories* and *sub-tasks*.

Of course, there are dependencies between stories and sub-tasks, but this has been abstracted

from the report to avoid confusion. The purpose of the Project Management section is simply to provide an outline of how the project is being managed.

4.1 Rough MVP Backlog

The *backlog* is a term to describe a list of tasks that will at some point need to be completed, in no particular order. This list is likely subject to change during the iterative process so a brief level of detail is used.

Legend: Stories are allocated a UID in the form **Ex-Sy**, with `x` corresponding to the Epic the story is part of, and `y` corresponding to the story number for that epic.

E1. Build the Initial Controller

- **E1-S1.** Decide on sensors
- **E1-S2.** Learn Particle cloud software and Command Line Interface (CLI)
- **E1-S3.** Collect sensor data through breadboard, incorporate sensor fusion
- **E1-S4.** Set up web-socket connections between microcontroller and laptop
- **E1-S5.** Solder components onto stripboard
- **E1-S6.** Attach electronics to wearable glove

E2. Recognise a Gesture

- **E2-S1.** Research and decide on candidates for gestures
- **E2-S2.** Set up data store for storing / loading data on server
- **E2-S3.** Collect sample data for training purposes
- **E2-S4.** Investigate HMMs
- **E2-S5.** Investigate Ensemble learning
- **E2-S6.** Investigate calibration for different user profiles

E3. Create a C2 Server for Interfacing the Controller to the Drone

- **E3-S1.** Integrate DJI SDK to control the drone
- **E3-S2.** Collect health metrics from both controller and drone
- **E3-S3.** Implement remote controller override
- **E3-S4.** Design and implement C2 server user interface
- **E3-S5.** Define and plan for all failure states
- **E3-S6.** Implement manual control mode (tilting hand moves drone)
- **E3-S7.** Define action sequences to be triggered by gestures

E4. Build the Second Controller Iteration

E4-S1. Consider other options for attaching to user's hand
E4-S2. Add battery power using voltage step-down breakout board
E4-S3. Add LED indicator
E4-S4. Investigate accessibility and inclusive design
E4-S5. Improve robustness of design
E4-S6. Test on breadboard
E4-S7. Final assembly

4.2 Risk Assessment

Upon reflection of the backlog, some tasks can be considered as higher 'risk' than others - the risk factor is determined based on the likelihood of complications arising and the student's existing experience with different aspects of the system. Higher risk stories should be expected to take longer than anticipated and this will need to be accommodated for.

4.2.1 High Risk Story Tasks

- **E2-S4** Investigate HMMs
- **E2-S5** Investigate Ensemble learning
- **E4-S2** Add battery power using voltage step-down breakout board

Note that the completion of these stories will lead to further tickets being raised, which is why the progress plan is not a complete step-by-step schematic. Details of training and feature extraction will be decided following further investigation into ML.

4.3 Ethics, Legal & Safety Concerns

- **Storing information** - Each user will have their own calibration profile on the system, which will require storing their name to associate them with it. Explicit consent should be provided for this.
- **Battery Powering** - The ideal low-weight battery source would of course be a Lithium Polymer battery. However, due to their volatile nature this project will avoid using LiPos.

- **Drone Flying** - Drones should only be flown in a safe and private environment, which is unlikely to include any rooms on campus. Testing will most likely be done off-site, unless explicit consent can be provided for a room on campus, in which case a risk assessment will need to be filled out.

5 Implementation

Contents

5.1 First Prototype

Before heavy investigation into data analysis and feature extraction could begin, it was first necessary to extract intital data from a controller prototype. Figure 5.1 shows the first result, prior to being installed onto the glove. The 3 flex sensors will be attached to the thumb, index and middle fingers - though complications arose when considering that the flex sensor may need to 'slip' slightly across the surface of the glove when the fingers are being bent.

Nonetheless, data is successfully being collected and streamed wirelessly. Figure 5.2 shows the received data on the server side, in a tuple consisting of the 3 flex sensor readings followed by pitch, roll and yaw (in that order). The resistances of the flex sensors are read through the Photon's analog pins, and reported in the range 0-1023.

For streaming of sensor data, UDP was chosen for improved latency. The tradeoff of having the occasional corrupted or undelivered packet here should not drastically affect the performance of the device. Contrastly, a *separate* Transmission Control Protocol (TCP) socket is used for both initialising the connection and maintaining health metrics. The ESTOP command, when implemented, will also be sent across TCP in order to guarantee its delivery.

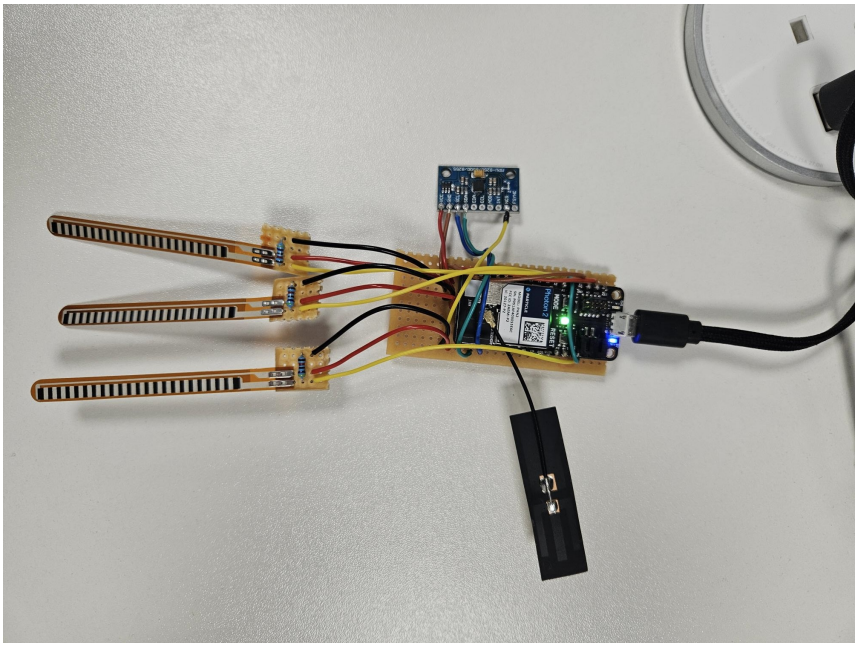

Figure 5.1: Initial prototype consisting of 3 flex sensors and an IMU

The initial wireless connection of the Particle Photon 2 was unstable, and so an antenna was added, which proved more reliable. This will of course have power implications when considering how much current the device will need to draw. Following discussion with the Supervisor, powering the device via a USB cable will be sufficient for initial data collection - though future versions should be battery powered in order to allow unrestricted movement and more accurately depict gestures.

The chosen technique of Sensor Fusion for this prototype is a Mahony filter, implemented to fuse the measurements of the accelerometer, gyroscope and magnetometer to obtain reliable pitch, roll and yaw readings. As this is a solved problem, library code was taken from an existing implementation [26]. Attempts were made at reading from the DMP of the device, but were not successful - either way, having the Mahony filter parameters available for tweaking may prove useful down the line.

```
Received from 192.168.1.45: FLEX SENSORS: 990, 306, 299 | PITCH: 12.07409, ROLL: -65.75082, YAW: 31.63100
Received from 192.168.1.45: FLEX SENSORS: 990, 304, 298 | PITCH: 11.87069, ROLL: -66.16822, YAW: 32.34980
Received from 192.168.1.45: FLEX SENSORS: 990, 301, 300 | PITCH: 11.82679, ROLL: -65.92062, YAW: 32.23530
Received from 192.168.1.45: FLEX SENSORS: 990, 296, 301 | PITCH: 11.63059, ROLL: -65.97472, YAW: 32.30840
Received from 192.168.1.45: FLEX SENSORS: 990, 297, 302 | PITCH: 12.27079, ROLL: -65.63472, YAW: 32.45390
Received from 192.168.1.45: FLEX SENSORS: 990, 297, 298 | PITCH: 12.42769, ROLL: -65.62582, YAW: 32.38890
Received from 192.168.1.45: FLEX SENSORS: 990, 301, 299 | PITCH: 13.20459, ROLL: -65.46452, YAW: 32.35920
Received from 192.168.1.45: FLEX SENSORS: 990, 297, 305 | PITCH: 13.47429, ROLL: -65.50562, YAW: 32.74140
Received from 192.168.1.45: FLEX SENSORS: 990, 305, 301 | PITCH: 12.77379, ROLL: -65.16692, YAW: 32.51560
Received from 192.168.1.45: FLEX SENSORS: 990, 303, 300 | PITCH: 12.62599, ROLL: -64.83052, YAW: 31.92380
Received from 192.168.1.45: FLEX SENSORS: 990, 302, 306 | PITCH: 12.11559, ROLL: -64.62512, YAW: 31.90510
```

Figure 5.2: Controller streaming flex sensor resistances and filtered IMU data to the server

To summarise, **E1** is largely complete, with the exception of **E1-S6** of attaching the circuitry to a glove.

6

Evaluation Plan

Contents

6.1 NASA TLX

To evaluate the 'effectiveness' of a robotics controller, it is not enough to just observe the accuracy of the device. Other HRI metrics need to be taken into account, measuring the overall experience of the user. The objective of the project is to reduce cognitive load when using the controller. One such example of a metric to measure this is the NASA TLX [10], consisting of six factors:

- Mental Demand
- Physical Demand
- Temporal Demand
- Performance
- Effort
- Frustration Level

Each of these factors is evaluated on a 5-point Likert-scale (participants rate each factor from 1-5). This provides a deeper insight into the user's experience, and although it is subjective, when used with the same users over multiple tests it can quantify the difference between successive designs.

6.2 Experiments

In order to evaluate the efficiency of the controller, the study will incorporate three practical tests:

- **Manual Mode Accuracy Test**: The users will be asked to fly the drone through a small obstacle course using manual mode, consisting of 3 rings to fly through. Two metrics will be collected: The number of collisions, and the time taken to complete the course. Though the *Tello* drone is quite harmless, the appropriate safety measures will of course need to be taken.
- **Gesture Mode Accuracy Test**: The users will play a game that lasts 1 minute. During the game, a random gesture will flash on the screen and the user will need to make that gesture with the controller in order to gain a point. The objective is to make as many correct gestures as possible. The reason for a time limit is to see how a participant's gestures might vary when under pressure.
- **False Positive Test**: The user will be asked to perform various physical tasks (including solving a jigsaw puzzle and performing 10 starjumps, this is still TBC), to measure the number of false positives recorded by the controller. The aim is to minimise the number of gestures recognised during everyday activities.

Bibliography

[1] DJI, *Tello product page*, https://store.dji.com/uk/product/tello?vid=38421, 2025.

[2] R. Robotics, *Tello sdk documentation*, https://dl-cdn.ryzerobotics.com/downloads/tello/20180910/Tello%20SDK%20Documentation%20EN_1.3.pdf, Oct. 2018.

[3] R. Robotics, *Tello sdk 2.0 user guide*, https://dl-cdn.ryzerobotics.com/downloads/Tello/Tello%20SDK%202.0%20User%20Guide.pdf, Nov. 2018.

[4] DJI, *Tello-python*, https://github.com/dji-sdk/Tello-Python, Feb. 2019.

[5] J. Willoughby, *Hands-on with the ryze tello*, Aug. 2024. [Online]. Available: https://www.heliguy.com/blogs/posts/hands-on-with-the-ryze-tello/.

[6] DJI, *Dji mavic 3 pro*, https://www.dji.com/uk/mavic-3-pro, 2024.

[7] DJI, *Dji system security white paper*, https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://djiarsmadrid.com/pdfdoc/DJI%2520Security%2520White%2520Paper.pdf&ved=2ahUKEwiwkpSPjv2KAxUPVUEAHbbOL7AQFnoECBMQAQ&usg=AOvVaw05YqguOd0q2aaxaawK0N8M, Apr. 2020.

[8] N. Schiller, M. Chlosta, M. Schloegel, *et al.*, "Drone security and the mysterious case of dji's droneid," in *Network and Distributed System Security (NDSS) Symposium 2023*, Jan. 2023. DOI: https://doi.org/10.14722/ndss.2023.24217.

[9] DJI, *Dji rc pro*, https://www.dji.com/uk/rc-pro, 2024.

[10] R. C. Quesada and Y. Demiris, "Design and evaluation of an augmented reality head-mounted display user interface for controlling legged manipulators," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 950–11 956. DOI: 10.1109/ICRA48891.2023.10161278.

[11] R. Chacón Quesada and Y. Demiris, "Multi-dimensional evaluation of an augmented reality head-mounted display user interface for controlling legged manipulators," *J. Hum.-Robot Interact.*, vol. 13, no. 2, Jun. 2024. DOI: 10.1145/3660649. [Online]. Available: https://doi.org/10.1145/3660649.

[12] D. Zhang, Y. Guo, J. Chen, J. Liu, and G.-Z. Yang, "A handheld master controller for robot-assisted microsurgery," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 394–400. DOI: 10.1109/IROS40897.2019.8967774.

[13] S.-O. Shin, D. Kim, and Y.-H. Seo, "Controlling mobile robot using imu and emg sensor-based gesture recognition," in *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications*, 2014, pp. 554–557. DOI: 10.1109/BWCCA.2014.145.

[14] G. Škulj, R. Vrabič, and P. Podržaj, "A wearable imu system for flexible teleoperation of a collaborative industrial robot," *Sensors (Basel, Switzerland)*, vol. 21, no. 17, p. 5871, 2021. DOI: 10.3390/s21175871.

[15] J.-W. Lee and K.-H. Yu, "Wearable drone controller: Machine learning-based hand gesture recognition and vibrotactile feedback," *Sensors*, vol. 23, no. 5, 2023, ISSN: 1424-8220. DOI: 10.3390/s23052666. [Online]. Available: https://www.mdpi.com/1424-8220/23/5/2666.

[16] N. Nguyen-Duc-Thanh, S. Lee, and D. Kim, "Two-stage hidden markov model in gesture recognition for human robot interaction," *International Journal of Advanced Robotic Systems*, vol. 9, p. 39, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:62564723.

[17] C. Tjhai and K. O'Keefe, "Using step size and lower limb segment orientation from multiple low-cost wearable inertial/magnetic sensors for pedestrian navigation," *Sensors*, vol. 19, p. 3140, Jul. 2019. DOI: 10.3390/s19143140.

[18] T. InvenSense, *Mpu-9250 register map and descriptions revision 1.6*, https://invensense.tdk.com/wp-content/uploads/2015/02/RM-MPU-9250A-00-v1.6.pdf, Jul. 2015.

[19] SparkFun, *Flex sensor data sheet*, https://docs.rs-online.com/8e1a/A700000011099021.pdf.

[20] Particle, *Photon 2 datasheet*, https://docs.particle.io/reference/datasheets/wi-fi/photon-2-datasheet/, 2018.

[21] A. Jensen, C. Coopmans, and Y. Chen, "Basics and guidelines of complementary filters for small uas navigation," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013, pp. 500–507. DOI: 10.1109/ICUAS.2013.6564726.

[22] D. Giaouris, *Chapter 3, Digital Control, EEE8007*. Apr. 2008. [Online]. Available: https://www.staff.ncl.ac.uk/damian.giaouris/pdf/Digital%20Control/Chapter3.pdf.

[23] K. Çoçoli and L. Badia, "A comparative analysis of sensor fusion algorithms for miniature imu measurements," in *2023 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2023, pp. 239–244. DOI: 10.1109/ISITIA59021.2023.10220994.

[24] Atlassian, *Agile best practices and tutorials*, 2019. [Online]. Available: https://www.atlassian.com/agile.

[25] M. Rehkopf, *Epics, stories, themes, and initiatives | atlassian*, 2019. [Online]. Available: https://www.atlassian.com/agile/project-management/epics-stories-themes.

[26] B. Wilkins, *Mpu-9250_breakout*, https://github.com/eliteio/MPU-9250_Breakout/tree/master, Jan. 2016.