HOLD MY COFFEE

Drinks Stabilization Platform

Project Members

Valentin Pletea-Marinescu *Software Development*
Faculty of Automatic Control and Computer Science

Sebastian-Alexandru Matei *Hardware Development*
Faculty of Engineering in Foreign Languages

Contents

List of Figures

1 Introduction

The transport of liquids is a common problem in everyday life. Whether at home or in the restaurant business, there is always a risk of spilling the drink while walking. This often occurs when the container is shaken and tilted by obstacles, bumps, slopes, steps, but also by our own movements. An example of this are restaurant employees who must often pass through very busy environments to take food or drinks to tables, so it is easy to make the glass spill, or a person who takes a drink from one room to another.

The purpose of the project Hold my Coffee was to implement a platform with an active stabilization function that, despite any movements of the base, keeps the upper surface in a stable horizontal position and does not allow the liquid to spill. We based the project on a similar known stabilization mechanism and built on existing models and solutions. The work is based on existing works on stabilizing platforms [1][2].

The system was built with a mechatronic approach and based on real-time inertial measurements to counteract the changes in the tilt angle. It features three degrees of freedom and thus stabilizes itself in both angular (pitch and roll) and vertical directions. It can counteract disturbances using its position control loop, which results in motion in these three directions [3].

The implementation was based on three high-performance brushless motor controllers with integrated FOC, Inertial Measurement Units (IMUs) with built-in sensor fusion [4], and two microcontrollers with real-time control loops that communicate with each other to achieve ultra-low control latencies.

In the project, we set ourselves the following goals: to create a stabilization algorithm that would counteract pitch and roll in real-time with a cascaded PID approach [5][6], implement height regulation, an independent from stabilization height control for the platform, that would keep it at a desired distance from the base and would use time-of-flight distance sensor for distance measurements [7], setup multiple communication protocols and implemented their communications, like FDCAN and UART, for synchronous data sharing between two processors for coordinated multi-threaded control operation [8], as well as to create a hardware design that allows a quick swap of components to create a flexible platform for further work and improvements.

For the stabilization algorithm, we are using quaternion estimation to track platform orientation to avoid singularities [9]. The estimation used a simple complementary filter [10] to fuse angular rate measurements from the gyroscope and acceleration measurements from the accelerometer.

For the cascade PID, we use two PID controllers: the position controller runs on the main microcontroller, while the motor controllers also provide high-performance current/velocity loops [11], resulting in cascaded multi-loop disturbance rejection.

The developed stabilization platform can have many potential use cases. We have previously discussed some examples. It can also be used, for instance, for transport of medication or other objects by medical staff, where it is important that they do not fall out of a container and do not become contaminated, or as a carrying platform for drones and other devices [12], or a mobile robot, where it is necessary to stabilize a surface on rough terrain or slopes [13]. The stabilization system shows that high-cost engineering solutions can be developed for less expensive and smaller devices using accessible components and open-source design tools.

2 Current State of the Field

Active stabilization has been applied in various domains, and some of the corresponding techniques can be leveraged in our platform. This section describes current state of gimbal stabilization and related control systems. The description of various stabilization systems in the literature highlights the features of the Hold my Coffee project.

2.1 Camera Gimbal Systems

Camera stabilization represents one of the most advanced implementations of the active gimbal. Typical three-axis gimbals, such as products by DJI and Zhiyun, use brushless motors controlled by a separate microcontroller and advanced motion algorithms to provide sub-degree stabilization [1]. The systems use inertial measurement units (IMU) with update rates of over 1000 Hz to measure the angles and angular velocities. The control algorithms usually consist of two cascaded PID controllers, where the outer loop controls the orientation and the inner loop provides velocity control [3][14]. Our system can be seen as an implementation of similar principle on a different load type. The main difference between the camera gimbal system and our system is that in our system, the center of mass of the payload is dependent on the amount of beverage in the cup. This will require different set of control parameters, and re-tuning of the controller may be required for each change in fill-level.

2.2 Tank Stabilization Systems

The military industry has developed several solutions for stabilizing the orientation of a weapon platform on a moving vehicle [2]. The most famous example is the video where the German Leopard tank keeps a beer glass stable on the tank gun while driving and turning rapidly. The system uses hydraulic motors for rotating the platform and a gyroscopic sensor to measure the relative orientation on three axes. The stabilization system is an example of three-axis active gimbal. Our system is conceptually similar, but at a much smaller scale, and it uses electrical motors instead of hydraulics. In this system, the control loop does not have to be very fast as the goal is to maintain orientation, rather than acceleration. Therefore, tank stabilization systems have very high latency in response, as compared to our beverage stabilization system. While large orientation change in the tank stabilization system would lead to catastrophic effects, e.g. the glass would fall on the ground. However, in our project a faster response will help prevent liquid from spilling from the cup, due to lower latency.

2.3 Ship Stabilization Systems

Ship-borne stabilization systems have a similar goal, in that they need to maintain a level platform under continuous environmental disturbance. Typical modern implementations are either passive (gyroscopes) or active (fin-based systems that adjust their position to counteract waves). For ship stabilization systems, restaurants and luxury cruise liners have also started installing table mounted stabilizers to provide a smooth dining experience to customers. Ship stabilization systems have lower bandwidth than camera gimbals, as their goal is to counteract relatively slow disturbance with lower peak accelerations than a gimbal might have to provide. However, ship stabilization systems have a much larger load than camera gimbals, or our stabilization system.

2.4 Robot Platform Stabilization Systems

Recent years of mobile robotics research have resulted in several self-balancing platforms and mobile robots, such as the two-wheel inverted pendulum robots, or more complex quadruped robots with torso stabilization [13][15]. Projects such as Boston Dynamics' Spot robot has demonstrated impressive stabilization during locomotion, which has been made possible by sensor fusion and predictive control. These systems are different from our problem as they are designed for the purpose of keeping the robot stable rather than a separate load. We are essentially inverting the control problem, where instead of keeping the robot balanced, the robot is treated as disturbance source.

2.5 Disturbance Rejection Control

In recent years, there have been several works that have explored different methods of disturbance rejection for inertially stabilized platforms [16][17]. These works have shown considerable improvements in performance

over PID controllers. A common theme in these works has been to apply active disturbance rejection control (ADRC) with some noise reduction observer, in order to estimate the unknown disturbances and reject them.

2.6 Orientation Estimation

Orientation estimation plays a key role in stabilization systems. Recent works have focused on the quaternion-based Kalman filtering for sensor fusion in orientation estimation [4][10]. A complementary filter has also been shown to have state-of-the-art performance for orientation estimation [9][18].

2.7 Limitations of Existing Platforms

To the best of our knowledge, there do not exist any stabilization platforms that are available commercially and can be easily used for beverage transportation. Products that exist tend to fall into one of two categories - either passive mechanical dampeners that provide very limited functionality for countering sudden disturbances or very expensive professional systems that are at price points much higher than consumer products. Academic projects have tried to solve the same problem in the past, but the implementations typically lack in terms of power-efficiency, form-factor, and ease-of-use considerations.

2.8 Features of Our System

Hold my Coffee platform has several characteristics that differentiate it from existing solutions. These are: First, the platform uses a dual-microcontroller architecture with separate units to process the high bandwidth (motor control) and low bandwidth (sensor fusion) signals. The separation allows the platform to use complex software that would be difficult to run on a single microcontroller. Second, the platform provides not only angular stabilization but also stabilization on the vertical axis. This addresses a problem with many existing gimbal-based stabilization systems that do not provide any mechanism to control the height of the payload. Third, we have used advanced brushless motor controllers that provide torque-mode operation and have integrated position sensing. This simplifies the hardware design and enables us to avoid the use of additional external encoders for position sensing. Finally, the use of modular 3D printed parts and aluminum extrusion profiles enables easy customization and repair as compared to commercial systems.

3 Hardware Resources

The platform has multiple hardware subsystems that have been chosen to satisfy specific requirements of the stabilization system. The section below describes the hardware components and how they fit in the overall system.

3.1 Microcontrollers

3.1.1 Teensy 4.1

The primary microcontroller used in the system is a Teensy 4.1 development board that contains an NXP i.MX RT1062 ARM Cortex-M7 processor that runs at 600 MHz. This microcontroller is used as the main processor and runs the pitch and roll stabilization software while also handling the FDCAN communication with the motor controllers. The Teensy 4.1 board was chosen as it natively supports FlexCAN and we needed to use FDCAN for communication with motor controllers [8]. The Teensy 4.1 also has sufficient processing power to run quaternion orientation calculations [9][18] and control algorithms in real-time.

3.1.2 Arduino UNO

The secondary microcontroller used in the system is an Arduino UNO that is based on an ATmega328P microcontroller. The Arduino UNO board is used to control two servo motors which provide a second layer of

stabilization for the lower platform. It also interfaces with a VL53L8CX time-of-flight distance sensor for sensing the distance [7]. The choice of a second microcontroller for this task is both architectural as well as practical. First, it is possible to control FDCAN, IMU and Servo motors on a single microcontroller, but doing so would lead to timing constraints that would have resulted in system not being responsive enough. Second, a dual-controller architecture would allow independent development of the two subsystems, and it would be much easier to debug if needed.

3.2 Motor Controllers and Motors

3.2.1 Moteus r4.11 Controllers

The primary actuation system is controlled using three Moteus r4.11 motor controllers that drive the brushless gimbal motors. Moteus controllers are a new class of brushless motor controller designed by mjbots. Moteus controllers provide position, velocity, and torque control for brushless motors and have an integrated current sensor and encoder processing and FOC (Field-oriented control) algorithms built-in [14]. The r4.11 version has 24V operation and higher current ratings than previous iterations, and is also thermally more stable during extended operation.

Moteus controllers communicate using the FDCAN protocol [8] and the low latency of CAN bus allows us to query all the controllers as well as send commands in parallel, allowing us to synchronize multi-axis control. We only use the position control mode and hence we treat the Moteus as a position controller with integrated position sensing for each of the motors. The Moteus controller also has an internal position loop and velocity loop, so our controller only needs to provide position setpoints to the motor controllers.

The Moteus controllers provide cascaded PID control with two control loops [11] - pid_dq loop (low-level current control) which controls the motor torque, and pid_position (high-level position control loop) which tracks the position. The default control parameters of these control loops are set using the FDCAN interface with help of the dedicated software tools. The values of control parameters have been set individually for each motor in our system as they have different mechanical load and dynamic behavior.

3.2.2 mj5208 Brushless Motors

The motors that are used for actuation in the primary platform are three mj5208 brushless gimbal motors. The motors are direct-drive (no reduction gearbox), pancake style with matching Moteus controllers. The motors provide sufficient torque for the stabilization task at hand while also being low in inertia for better system responsiveness to sudden disturbances. Two of the three motors are used for controlling the pitch and roll axis. The third motor is used to drive a lead screw mechanism that moves the entire platform in the vertical direction.

3.2.3 Servo Motors

The secondary layer of the stabilization system on the lower platform consists of two hobby-grade servo motors that are controlled by the Arduino UNO. This hierarchical setup where the primary platform driven by Moteus stabilizes the load followed by the lower servo-based platform taking care of smaller disturbances prevent the motors from saturating, even during large disturbances.

3.3 Sensors

3.3.1 BNO055 Inertial Measurement Unit

The sensor used for orientation sensing is an Adafruit BNO055 absolute orientation sensor. It is a combination of a 3-axis accelerometer, 3-axis gyroscope, and a 3-axis magnetometer with a built-in sensor fusion algorithm. The sensor can output orientation as quaternions, Euler angles, rotation matrices, etc. This reduces the need for any computationally intensive sensor fusion algorithms on the microcontroller. We have used the sensor in

NDOF (Nine Degrees of Freedom) mode which fuses data from all the sensors and provides drift-compensated orientation estimates using a complementary filter [9][18].

The sensor is mounted rigidly to the stabilized platform and we can use the orientation data provided by it to provide feedback to our control algorithm. The BNO055 is connected to the Teensy 4.1 over I2C bus at 400 kHz, providing us with about 100 Hz update rate. The update rate is more than sufficient for our required control bandwidth and still much lower than what the Teensy 4.1 is capable of.

3.3.2 VL53L8CX Time-of-Flight Sensor

The sensor used for vertical position measurement is an Adafruit VL53L8CX multi-zone time-of-flight distance sensor mounted on the underside of the platform. This sensor is a generational improvement over previous generation ToF sensors, with 8x8 zone ranging with improved accuracy and range [7]. The VL53L8CX works by modulating infrared light and measuring the phase shift of reflected light to estimate the distance with sub-mm accuracy in multiple zones.

VL53L8CX works with a measurement range of 10 mm to 4000 mm which is more than what we require for our vertical adjustment mechanism. For our application, we are only using single-zone mode to obtain a distance measurement in the center, though the multi-zone capabilities could be used to extend the functionality of our system to include tilt-detection, object avoidance, etc.

The VL53L8CX is connected to the Arduino UNO over I2C bus, and the distance measurements are used as feedback to the vertical position control loop which drives the lead screw motor to maintain the platform at a target height.

3.4 Power System

The system is powered by an HRB lithium-polymer battery pack with 6000 mAh capacity, 14.8V nominal voltage (4S) and continuous discharge rating of 50C. This battery has high enough discharge rate that it can provide the required current during simultaneous motor actuation, in particular when all three axes are correcting a large disturbance. 50C rating implies a continuous discharge current of 300A which is much larger than the system peak current requirements and also ensures good thermal stability and battery life. The battery is connected to the system through an XT60 connector that can handle the peak currents with low resistive losses.

3.4.1 Power Protection and Safety

The power system incorporates multiple layers of protection to ensure safe operation during demonstrations and regular use. A waterproof 300A ignition-protected switch serves as the main system power control, functioning as a push-button activation mechanism. This switch is positioned between the battery and the power distribution system, allowing for immediate system shutdown in emergency situations.

The electrical protection includes automotive-grade 300A fuses that provide short-circuit protection for the entire system. These fuses are strategically placed in the power distribution path to protect both the battery and downstream components from overcurrent conditions that could result from wiring faults or component failures.

Low voltage monitoring is implemented to protect the lithium-polymer battery from over-discharge, which can cause permanent damage to the cells. The system generates audible alarms when the battery voltage drops below safe operating thresholds, alerting the operator to recharge or replace the battery before critical voltage levels are reached.

During demonstrations, additional safety protocols are followed: the battery is stored in a fire-resistant LiPo safety bag when not in use, all connections are visually inspected before powering the system, and the emergency shutdown switch remains easily accessible to the operator at all times.

Separate buck converter modules are used for logic-level parts and motor controllers. A buck converter module steps down the battery voltage to 5V for powering the Teensy 4.1 and Arduino UNO microcontrollers.

This switching regulator is highly efficient over the entire input voltage range as the battery discharges, and will not get too hot compared to linear regulation. Buck converter also has tight output voltage regulation ensuring stable operation of sensitive digital logic and sensor circuits.

The Moteus controllers are connected directly to the battery voltage as they have integrated regulation and protection circuits and are rated to be directly connected to a multi-cell lithium battery. This direct connection reduces resistive losses in the power path and also allows the motor controllers to directly and efficiently drive the motors using their integrated field-oriented control algorithms [14].

3.5 Mechanical Structure

The overall platform structure is made up of 3D-printed parts using ABS (Acrylonitrile Butadiene Styrene) and PETG (Polyethylene Terephthalate Glycol) thermoplastics along with aluminum extrusion profiles which provide the vertical support frame. ABS has excellent dimensional stability and impact resistance for structural parts while PETG has better layer adhesion and flexibility for parts with dynamic loads. Custom motor mounts are also 3D-printed through fused deposition modeling to mount the mj5208 motors on the platform and ensure that they are properly aligned. The platform surface also has a silicone mat which provides friction between the mat and any beverage containers to prevent them from sliding on the surface.

Aluminum extrusion profiles provide the vertical support frame for the entire system, and connect the base to the stabilized platform while providing mounting points for mechanical subsystems. The choice of extrusions was driven by their high strength to weight ratio, modular T-slot mounting system, and availability of various cross-sections. The T-slot mounting patterns also allow for parts to be attached easily without custom machining.

The vertical actuation is provided using an ACME T8 lead screw driven by the third mj5208 motor. The T8 means that it is an 8mm nominal diameter lead screw with a trapezoidal thread profile. The lead screw converts rotary motion from the motor into linear motion of the platform and the thread-pitch provides the mechanical advantage and the positioning resolution. Two MGN linear rail assemblies are used to guide the platform in the vertical direction. This prevents rotation and binding during vertical actuation of the platform. The miniature linear guides have very low friction, with recirculating ball bearings on a machined rail profile providing precise constraint.

The mechanical design of the platform has been kept modular with all the component mounting patterns being such that it allows individual parts to be replaced without having to disassemble the entire platform. The modular design philosophy is followed throughout the system right from the motor mounting brackets to sensor housings.

Fig. 1 and Fig. 2 present comprehensive views of the complete mechanical assembly, illustrating the integration of 3D-printed components with aluminum extrusions, linear guidance systems, and the lead screw mechanism.

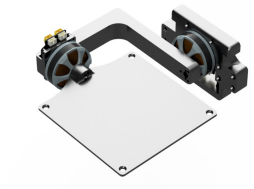

Figure 1: Mechanical assembly overview showing the integration of stabilized platform and motor mounts

Figure 2: Mechanical assembly highlighting the MGN linear rails, T8 lead screw, and 3D-printed structural components fabricated from ABS and PETG.

4 Software Resources

A brief introduction to the various software packages and libraries used in the development and deployment of the control algorithms and hardware interfaces on the Hold my Coffee platform.

4.1 Development Environments

The Teensy 4.1 firmware was developed in the Arduino IDE with Teensyduino add-on, which provides Teensy-specific libraries and compilation target. The Arduino framework provides a high level of abstraction from the lower level hardware, while still giving enough low level access when needed. At the same time, it provided a rapid prototyping environment as well as easy access to abundant Arduino community resources.

The Arduino UNO firmware was developed in the stock Arduino IDE with no extensions.

Git was used for the version control system during the project. It provided a decentralized control, which allowed us to work on different parts of the system independently, and only merge together when needed.

4.2 Communication Protocols

4.2.1 FDCAN Protocol

FDCAN is a modified version of the CAN (controller area network) protocol which provides flexible data rate option. The data rate can be increased during data phase of frame transmission but arbitration phase of FDCAN still remains backward compatible with CAN. FDCAN is the main form of communication between Teensy and the Moteus controllers [8].

The bus was configured for 1Mbps nominal bit rate and a data phase multiplier to allow a higher effective data rate and provide 100Hz control loop for each of the 3 motors.

FDCAN is configured in query response mode, where Teensy issues commands and immediately receives the response from each of the controllers. This allows tight binding between command and response.

4.2.2 UART

Teensy and Arduino are connected using a UART (universal asynchronous receiver/transmitter) serial interface configured at 115200 baud. Teensy regularly sends the pitch and roll angles over the serial interface to the Arduino, and Arduino uses those angle values to drive secondary stabilization servos. In the opposite direction, Arduino responds to request from Teensy to provide distance readings by sending VL53L8CX data, and Teensy uses those measurements to control the vertical position.

The bi-directional UART communication uses a simple text protocol, with both sides sending and receiving human-readable strings with each message terminated by newline character. This is not an efficient communication protocol, but made debugging easier by making it possible to read messages directly in a serial terminal.

4.2.3 I2C

Teensy and Arduino use I2C bus to communicate with the IMU and the distance sensor, respectively. I2C is a two-wire interface, where the microcontroller acts as master and each of the sensors act as slaves. The I2C buses are both set to 400kHz fast-mode.

5 Hardware Implementation

The description of the construction methods and the working principles of different components of the Hold my Coffee platform.

5.1 System Block Diagram

Fig. 3 presents the complete system architecture, illustrating the interconnections between all hardware and software components. The diagram shows the dual-microcontroller architecture with the Teensy 4.1 serving as the primary controller for angular stabilization and the Arduino UNO handling height control and secondary servo actuation.

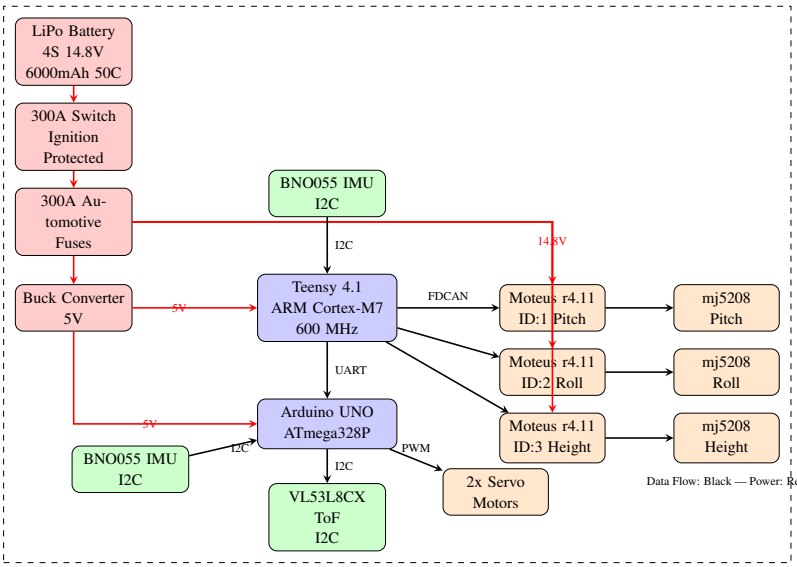

Figure 3: System block diagram showing dual-microcontroller architecture, power distribution, and communication buses between all components

The power system architecture provides multiple protection layers with the 300A waterproof ignition-protected switch serving as the main activation control. The 300A automotive fuses protect against short-circuit conditions, while the buck converter provides regulated 5V power to the microcontrollers, isolating them from motor switching noise. The Moteus controllers receive unregulated battery voltage directly, utilizing their integrated power management for efficient motor drive.

Communication architecture employs three different protocols optimized for their respective data rates and latency requirements. FDCAN operates at 1Mbps nominal rate for high-bandwidth motor command and telemetry exchange with the three Moteus controllers. UART at 115200 baud facilitates inter-processor communication between Teensy and Arduino for angle data and distance measurements. I2C at 400kHz provides sensor interfaces for both the BNO055 IMU and VL53L8CX distance sensor.

5.2 Mechanical Subsystems

5.2.1 Platform

The upper stabilization platform was 3D printed in two pieces, each around 200x200mm square. This provided a mounting surface for the mj5208 motors, and mounting points for the BNO055 sensor. We also added bosses to the printed part around the motor mounting holes. The boss increases the number of thread engaged in the plastic to improve the holding power of the screws. The top surface of the platform had a silicone sheet attached to it with a pressure-sensitive adhesive. This increases the friction between the sheet and a cup to prevent the cup from sliding when the platform is rotating.

Pitch and roll motors were mounted on two adjacent sides, and the height motor was mounted at the center. This keeps the distance between the motors and gimbal rotation points as small as possible to reduce the torque required from motors. Motors were connected to each of the actuators with 3D-printed linkages. Pitch and roll motors are connected directly to the gimbal rings of the platform. Height motor is connected to the lead screw through the linkage.

5.2.2 Vertical Axis

The vertical actuation system consists of a T8 ACME lead screw, a nut, and linear rail assemblies to guide the motion. The vertical motor drives the T8 lead screw to move the platform up and down. T8 lead screw is 8mm in diameter, with 2mm pitch trapezoidal thread. The motor is connected directly to the shaft of the lead screw with a 3D printed flexible coupling. Flexible coupling absorbs any misalignment between motor and lead screw shaft while transmitting torque from motor to the screw. The lead screw passes through a lead nut that is rigidly mounted to the top of the platform, converting rotational motion to linear translation along the vertical axis.

The vertical motion is constrained and guided with two MGN12H linear rail assemblies. These are miniature linear guides, which use recirculating ball bearing inside a hardened steel rail to guide the motion with low friction and high load capacity, in a very compact size. The rails are mounted parallel to the lead screw axis on the aluminum extrusion, and the top platform is supported by linear bearing blocks that allow only the translation along one axis.

Pitch of 2mm per rotation of the lead screw is represented in software by the MM_TO_ROT constant (0.5 rotations per mm of linear travel). This constant was determined empirically by measuring the platform travel with known motor rotation. This required some calibration due to some mechanical play in the coupling and the loss of contact in threads. The total vertical travel was limited to about 40mm.

5.2.3 Bottom Platform

The lower platform provides a base for the entire system, and holds Arduino UNO, the servo motors, and the VL53L8CX distance sensor. It was constructed using a combination of 3D printed ABS and PETG parts, and aluminum extrusion profiles for high stiffness members. Servo motors were mounted perpendicular to the base, and their output shafts were driving a linkage that tilted the upper stabilization platform. This provides a secondary level of stabilization that moves slower than the main Moteus-based stabilization, and can be used to provide coarse corrections for slow changes.

5.3 Electrical Subsystems

5.3.1 Power System

Battery is connected to the system through an XT60 connector, which has a large current rating and provides good retention. Power is branched from this point to different parts of the system with a distribution board, which we custom built. Moteus controllers are powered by the unregulated battery voltage, as they have internal buck converters that step the voltage down to the appropriate levels for the logic circuits and gate drivers.

Teensy 4.1 and Arduino UNO are powered by the 5V buck converter. Isolating these from the direct connection to the motors removes noise from the motor switching from the analog front end of the IMU interface. The wire gauge was chosen for all the connections taking into account the current they carry, and the distance. 16AWG silicone wire was used for power connections to motors. 22AWG wire was used for all the logic level signals.

5.3.2 Communication Wires

FDCAN communication wires between Teensy and each of the Moteus controllers are twisted-pair. The bus is laid out in a linear topology with 120-ohm termination resistors at each end to avoid signal reflections that would occur at the end of the bus [8].

I2C bus connections are short runs of four-conductor cable, with SCL, SDA, power and ground. We kept the sensors as close as possible to the respective microcontrollers to reduce the cable length. This minimizes the effects of electromagnetic interference, and also reduces the capacitive load on the signals that degrades signal quality.

UART communication between Teensy and Arduino uses three wires, one each for transmit, receive, and a common ground. TX and RX pins are cross-connected between the two processors.

5.4 Sensor Placement

5.4.1 BNO055 Mount

BNO055 is mounted to the top platform with standoffs to keep the sensor level aligned with the platform frame. It is important to keep the sensor orientation right, so that the sensor axes are aligned with the frame of the platform. This is necessary to make sure that the pitch and roll that the BNO055 reports actually correspond to the pitch and roll of the platform. We chose not to use any adhesive to mount the sensor, to make it easier to replace the sensor if it is needed later.

5.4.2 VL53L8CX Placement

VL53L8CX sensor was mounted below the bottom platform, facing downwards towards the ground. We made sure to place the sensor to take into account its field of view. The sensor has a configurable field of view, and in single zone mode, it provides a very narrow measurement spot with minimal divergence from the center. We made sure that the measurement zone of the sensor is always within the footprint of the base throughout its entire vertical travel range. This is to avoid getting false readings from reflections from objects that are not part of the measurement area.

The sensor runs continuously, with Arduino triggering a measurement and reading the result at every iteration of the control loop. The VL53L8CX has advanced signal processing and multi-zone capability [7] that provides robustness to ambient lighting and surface texture variations compared to previous generations of time-of-flight sensors.

5.5 System Diagrams

5.5.1 Activity Diagram

The activity diagram in Fig. 4 shows the main control loop and decisions in the stabilization system. The diagram also shows the parallel processing between the pitch/roll stabilization and the height control, which is a result of having two separate microcontrollers.

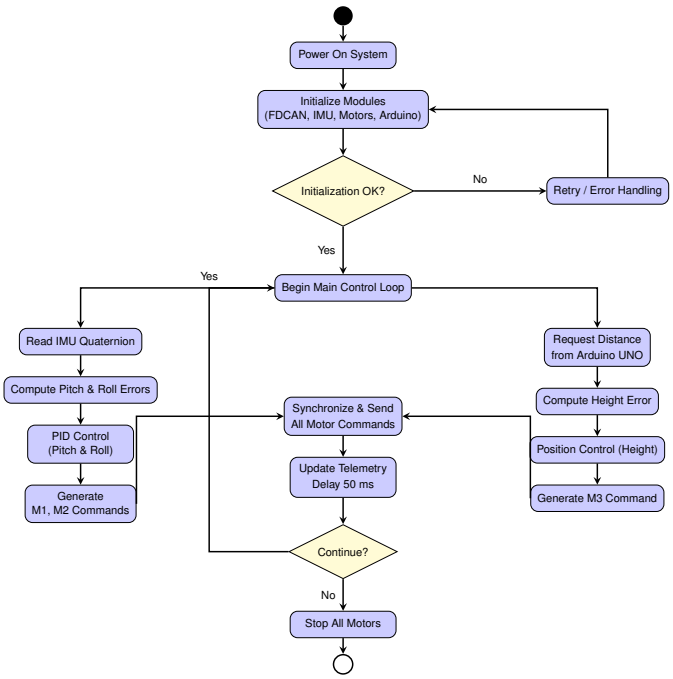

Figure 4: Activity diagram showing parallel IMU-based angular stabilization and height control processes converging into synchronized motor commands.

The activity diagram illustrates the main control loop structure, beginning with system initialization where FDCAN communication establishes connectivity with the Moteus controllers and the BNO055 sensor undergoes calibration. The runtime loop executes two parallel processing streams: angular stabilization using IMU feedback and height control using distance sensor data from the Arduino UNO.

5.5.2 Sequence Diagram

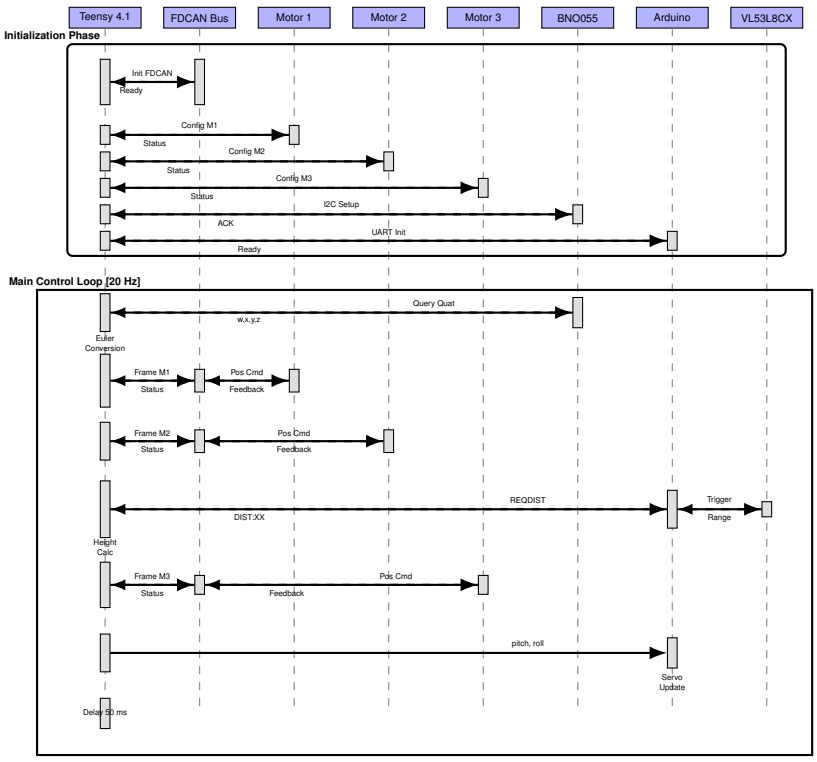

Figure 5: Sequence diagram illustrating communication timing and control flow between system modules.

The sequence diagram in Fig. 5 illustrates the temporal interaction between system components, showing message flow and timing relationships that characterize the distributed control architecture. The sequence diagram reveals the temporal coordination required for effective stabilization. Initialization proceeds serially, with the Teensy establishing communication with the Arduino via UART, configuring the BNO055 through I2C, and initializing the three Moteus controllers via FDCAN [8]. The main control loop demonstrates parallel query operations to the IMU and distance sensor, followed by near-simultaneous command transmission to all motors. This parallelism ensures that motor commands reflect contemporaneous sensor data, minimizing phase lag that would otherwise degrade control performance.

5.6 Laser Stabilization and Secondary IMU System

To ensure accurate distance measurements regardless of platform motion, a secondary sensing and stabilization unit was implemented on the laser sensor module. This subsystem is built around an **Arduino UNO** microcontroller that hosts a second **BNO055 IMU**, a **VL53L0X time-of-flight distance sensor**, and two **servo motors** arranged in an orthogonal configuration to control the pitch and roll of the laser assembly.

The second IMU continuously measures the orientation of the laser module relative to the global frame. Based on these readings, the two servo motors compensate for any tilt induced by platform movement, thus maintaining the laser sensor perpendicular to the ground. This mechanical stabilization ensures that the measured distance remains true even when the base or platform is moving.

6 Software Implementation

The implemented software realises the control strategies introduced above to keep the platform levelled regardless of motion of the base plate. This section details the mathematics and algorithms behind these methods using the form in which they are actually implemented. This includes both the high-level control code running on the Teensy and the PID loops inside each Moteus motor controller.

6.1 Cascaded Control Design

The control structure uses a cascaded controller with two levels [11]. The higher-level controller runs on the Teensy 4.1 at 20 Hz, determining the setpoint position for each motor based on the orientation measurement from the IMU. The lower-level controllers are implemented on each Moteus controller and operate at a much higher frequency of several kilohertz. They command the current and position of each motor to follow their respective setpoints.

6.2 Teensy Control Loop

The main Teensy control loop is in charge of reading the IMU sensor values, evaluating the control law, and sending motor commands. The individual steps of the loop are as follows: Read sensors, Calculate errors, Calculate outputs, Send motor commands, Delay. This loop runs with a frequency of 20 Hz, which is achieved by a 50 ms delay at the end of each iteration.

6.2.1 Conversion from Quaternion to Euler Angles

The orientation data from the BNO055 is in the form of quaternions, which are given as four scalar values (w, x, y, z). While the quaternion representation has several advantages, including the lack of singularities [10], it is not the most intuitive form to work with. For our controller, it is much more natural to think in Euler angles. We thus convert the quaternions to the roll and pitch angles in the following.

The conversion uses the standard equations that relate quaternions to the corresponding rotation matrix. For a quaternion $\mathbf{q} = [w, x, y, z]^T$, the roll angle ϕ (rotation around the x-axis) can be determined as:

$$\phi = \arctan 2(2(wx + yz), 1 - 2(x^2 + y^2)) \tag{1}$$

The pitch angle θ (rotation about the y-axis) requires special handling to avoid domain errors in the arcsin function:

$$\theta = \begin{cases} \text{sign}(\sin p) \cdot \frac{\pi}{2} & \text{if } |\sin p| \geq 1 \\ \arcsin(\sin p) & \text{otherwise} \end{cases} \tag{2}$$

where $\sin p = 2(wy - zx)$. This formulation prevents numerical instability near vertical orientations where the pitch angle approaches ± 90 degrees.Following computation in radians, the angles convert to degrees through multiplication by the factor $\frac{180}{\pi}$.

Implementation Code:

```
imu::Quaternion quat = bno.getQuat();
float w = quat.w(), x = quat.x(), y = quat.y(), z = quat.z();
float sinr_cosp = 2.0f * (w * x + y * z);
float cosr_cosp = 1.0f - 2.0f * (x * x + y * y);
float roll_rad = atan2(sinr_cosp, cosr_cosp);
float sinp = 2.0f * (w * y - z * x);
float pitch_rad = (abs(sinp) >= 1) ? copysign(M_PI / 2, sinp) : asin(sinp);
float roll_deg  = roll_rad  * 180.0f / M_PI;
float pitch_deg = pitch_rad * 180.0f / M_PI;
```

The implementation reads the quaternion components directly from the BNO055 sensor using the Adafruit library. The `atan2()` function handles all quadrants for roll computation without singularities. For pitch calculation, the conditional expression checks if `abs(sinp) >= 1` to detect near-vertical orientations. When detected, `copysign()` returns $+\pi/2$ or $-\pi/2$ with the correct sign, preventing numerical instability. Otherwise, the standard `asin()` function computes the pitch angle. The final conversion to degrees uses the constant `M_PI` from the math library.

6.2.2 Angular Error Calculation with Wraparound Handling

Computing orientation error requires careful handling of angular wraparound at the ± 180 degree boundaries. A naive subtraction of target from measured angle produces incorrect results when the angles span this discontinuity. The normalizeAngleDifference function addresses this through iterative reduction of the computed difference to the valid range:

$$\Delta\theta_{k+1} = \begin{cases} \Delta\theta_k - 360 & \text{if } \Delta\theta_k > 180 \\ \Delta\theta_k + 360 & \text{if } \Delta\theta_k < -180 \\ \Delta\theta_k & \text{otherwise} \end{cases} \tag{3}$$

This wraparound handling is critical for the roll axis, where the target angle is set to -172.61, very close to the -180 boundary. Without proper normalization, a measured roll angle of $+170$ would produce an error of -342.61 instead of the correct $+17.39$, causing the controller to rotate in the wrong direction.

Implementation Code:

```
float normalizeAngleDifference(float target, float current){
  float diff = target - current;
  while(diff > 180.0f) diff -= 360.0f;
  while(diff < -180.0f) diff += 360.0f;
  return diff;
}
float pitch_error = TARGET_PITCH - pitch_deg;
float roll_error  = normalizeAngleDifference(TARGET_ROLL, roll_deg);
```

The function uses while loops to iteratively reduce the angular difference into the valid range of $[-180, +180]$. This approach handles cases where the initial difference might be larger than 360 (though this rarely occurs in

practice). For pitch, since the target is -1.82, simple subtraction suffices. For roll with target -172.61, the normalization is essential to prevent control direction errors near the ± 180 boundary.

6.2.3 PID Control Law

The high-level control law on the Teensy implements full PID feedback with saturation limits to prevent actuator overdriving. For both pitch and roll axes, the correction command computes according to:

$$u_{\text{pitch}} = \text{sat}\left(K_{p,\text{pitch}} \cdot e_{\text{pitch}} + K_{i,\text{pitch}} \cdot \int e_{\text{pitch}} dt + K_{d,\text{pitch}} \cdot \frac{de_{\text{pitch}}}{dt}, -u_{\max}, +u_{\max}\right) \tag{4}$$

$$u_{\text{roll}} = \text{sat}\left(K_{p,\text{roll}} \cdot e_{\text{roll}} + K_{i,\text{roll}} \cdot \int e_{\text{roll}} dt + K_{d,\text{roll}} \cdot \frac{de_{\text{roll}}}{dt}, -u_{\max}, +u_{\max}\right) \tag{5}$$

where the PID gains are: $K_p = 0.10$, $K_{i,\text{pitch}} = 0.005$, $K_{i,\text{roll}} = 0.002$, $K_{d,\text{pitch}} = 0.004$, $K_{d,\text{roll}} = 0.0012$, and $u_{\max} = 8$ limits the maximum correction magnitude.

The integral term accumulates error over time to eliminate steady-state error, while the derivative term provides damping to reduce oscillations. The integral is clamped to ± 50 to prevent integral windup during saturation.

Implementation Code (Pitch Axis):

```
unsigned long now = micros();
float dt = (prev_time > 0) ? (now - prev_time) / 1e6f : 0.002f;
prev_time = now;
float pitch_error = TARGET_PITCH - pitch_deg;
pitch_integral += pitch_error * dt;
pitch_integral = constrain(pitch_integral, -50.0f, 50.0f);
float pitch_derivative = (pitch_error - pitch_prev_error) / dt;
pitch_prev_error = pitch_error;
float pitch_correction_deg = (PITCH_KP * pitch_error) +
                             (PITCH_KI * pitch_integral) +
                             (PITCH_KD * pitch_derivative);
pitch_correction_deg = constrain(pitch_correction_deg, -8.0f, 8.0f);
if (moteus_pitch.SetQuery()) {
    float pitch_correction_rad = pitch_correction_deg * M_PI / 180.0f;
    float current_pitch_pos = moteus_pitch.last_result().values.position;
    float target_pitch_pos = current_pitch_pos - pitch_correction_rad;
    target_pitch_pos = current_pitch_pos +
        constrain(target_pitch_pos - current_pitch_pos, -0.05f, 0.05f);

    Moteus::PositionMode::Command cmd;
    cmd.position = target_pitch_pos;
    cmd.maximum_torque = MAX_TORQUE;
    moteus_pitch.BeginPosition(cmd);
}
```

The implementation computes the time delta `dt` using `micros()` for precise timing at microsecond resolution. The integral term accumulates using discrete integration (`error * dt`), and `constrain()` prevents windup by clamping to ± 50. The derivative is computed from consecutive error samples to provide noise immunity. The final correction is saturated to ± 8 before conversion to radians. The motor command includes rate limiting (`constrain(..., -0.05f, 0.05f)`) to prevent abrupt position steps that would cause jerky motion. The same logic applies to the roll axis with different gain constants. The `MAX_TORQUE` constant of 7.0 Nm limits the maximum motor torque for safety.

6.3 Embedded Motor Controller PID Tuning

Each Moteus r4.11 controller implements two cascaded PID control loops [11]: a low-level current control loop (pid_dq) operating in the direct-quadrature reference frame for field-oriented control, and a higher-level position control loop (pid_position) that tracks commanded angular positions. These parameters were configured via

FDCAN interface using specialized tools, with motor-specific tuning to account for varying mechanical loads and dynamic characteristics.

Motor	ID	Current Loop (pid_dq)		Position Loop (pid_position)		
		$K_{p,dq}$	$K_{i,dq}$	$K_{p,pos}$	$K_{i,pos}$	$K_{d,pos}$
Pitch Motor	1	0.3	5.0	30.0	0.1	0.3
Roll Motor	2	0.3	2.0	15.0	0.2	0.1
Height Motor	3	0.03	59.7	4.0	1.0	0.05

Figure 6: PID controller parameters for all three motors showing current control loop and position control loop gains

The differences in PID parameters across the three motors reflect the distinct mechanical characteristics and performance requirements of each axis. The pitch and roll axes prioritize rapid disturbance rejection and stiff position holding, while the height axis emphasizes smooth, predictable motion. This motor-specific tuning, combined with the high-level PID control on the Teensy, creates a multi-layer control architecture that effectively stabilizes the platform across varied operating conditions.

6.3.1 Motor Position Command Generation

Moteus controllers accept position commands in units of motor shaft rotations. Converting angular corrections to position commands requires incorporating the current motor position, as commands specify absolute positions rather than incremental changes. The rate limiting prevents abrupt position steps that would cause jerky platform motion:

$$\theta_{\text{cmd}} = \theta_{\text{current}} + \text{sat}\left(\theta_{\text{target}} - \theta_{\text{current}}, -\dot{\theta}_{\max}, +\dot{\theta}_{\max}\right) \tag{6}$$

where $\dot{\theta}_{\max} = 0.05$ rad/cycle. Finally, the position command transmits via FDCAN along with the maximum torque constraint of 7.0 Nm.

6.4 Vertical Position Control

Height control maintains the platform at a target distance above the support surface, measured by the VL53L8CX sensor on the Arduino UNO. The control objective seeks to minimize the distance error:

$$e_{\text{height}} = d_{\text{target}} - d_{\text{measured}} \tag{7}$$

The error converts to motor position commands through a calibrated scaling factor:

$$\theta_{\text{motor3}} = \text{sat}\left(e_{\text{height}} \cdot k_{\text{screw}}, \theta_{\min}, \theta_{\max}\right) \tag{8}$$

where $k_{\text{screw}} = 0.5$ rotations/mm, and $\theta_{\min} = -40.0$, $\theta_{\max} = 0.5$ limit the travel range.

To reduce noise from the distance sensor, a moving average filter is implemented with a buffer size of 5 samples. This filtering smooths out measurement variations while maintaining sufficient responsiveness for height control.

Implementation Code:

```
#define FILTER_SIZE 5
float distance_buffer[FILTER_SIZE];
int buffer_index = 0;
bool buffer_filled = false;
void addDistanceSample(float new_sample) {
  distance_buffer[buffer_index] = new_sample;
  buffer_index = (buffer_index + 1) % FILTER_SIZE;
  if (buffer_index == 0) buffer_filled = true;
```

```
}
float getAverageDistance() {
  int count = buffer_filled ? FILTER_SIZE : buffer_index;
  if (count == 0) return distance_from_uno;
  float sum = 0.0f;
  for (int i = 0; i < count; i++) sum += distance_buffer[i];
  return sum / count;
}
if (Serial1.available()) {
    String msg = Serial1.readStringUntil('\n');
    msg.trim();
    if (msg.startsWith("DIST:")) {
      float val = msg.substring(5).toFloat();
      if (val > 0.0f && val < 10000.0f) {
        distance_from_uno = val;
        addDistanceSample(val);
      }
    }
}
float avg_distance = getAverageDistance();
if (moteus_height.SetQuery()) {
    float error_mm = TARGET_DISTANCE - avg_distance;
    float target_pos;
    if (fabs(error_mm) > 5.0f) {
      target_pos = constrain(error_mm * MM_TO_ROT, MIN_POS, MAX_POS);
    } else {
      target_pos = moteus_height.last_result().values.position;
    }
    Moteus::PositionMode::Command h_cmd;
    h_cmd.position = target_pos;
    h_cmd.maximum_torque = MAX_TORQUE;
    moteus_height.BeginPosition(h_cmd);
}
```

The moving average filter uses a circular buffer of 5 samples. The `addDistanceSample()` function adds new measurements using modulo arithmetic for wrap-around. The `getAverageDistance()` function computes the mean, handling the initial fill phase when fewer than 5 samples are available. Distance data arrives via UART from the Arduino as text messages formatted as "DIST:xxx". The `String.startsWith()` and `substring()` methods parse the value, with range checking to reject invalid measurements. The height control implements a 5mm deadband using `fabs(error_mm) > 5.0f`, preventing unnecessary actuation from sensor noise. When outside the deadband, the error scales by `MM_TO_ROT = 0.5` (rotations per mm) and saturates to the valid travel range.

6.5 Timing Considerations and System Performance

The timing of the control loop influences the performance of stabilization in various ways. The current update rate of 20 Hz ($T = 50$ ms) was chosen as a trade-off between system capabilities and performance observations. In discrete-time control, the closed-loop bandwidth is roughly one order of magnitude below the sampling frequency, resulting in $f_{\text{BW}} \approx 2$ Hz.

As per the timing analysis, the execution time of a typical loop iteration is around 30 milliseconds, distributed as follows: Query and Euler conversion of the IMU data: ~ 8 ms, Transmission and reception to/from the Arduino via UART: ~ 5 ms, Generation of motor commands: ~ 2 ms, FDCAN transmission and reception: ~ 15 ms.

6.6 Laser Sensor Stabilization and Distance Computation

A secondary control loop operates independently on the Arduino UNO, using the second BNO055 IMU to stabilize the VL53L0X laser sensor via two servo motors controlling pitch and roll. The goal is to maintain the laser axis perpendicular to the ground, ensuring consistent distance measurements regardless of platform tilt.

The BNO055 provides real-time orientation data (pitch and roll) through Euler angles. These values directly command the two servo motors, with servo1 corresponding to roll and servo2 to pitch motion.

Implementation Code:

```
Servo servo1, servo2;
Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28);
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
float getRealDistance(float d_perp, float pitch_deg, float roll_deg) {
  float pitch_rad = pitch_deg * DEG_TO_RAD;
  float roll_rad  = roll_deg  * DEG_TO_RAD;
  return d_perp / (cos(pitch_rad) * cos(roll_rad));
}
void setup() {
  Serial.begin(115200);
  Wire.begin();
  servo1.attach(9);
  servo2.attach(10);

  if (!bno.begin()) while (1);
  bno.setExtCrystalUse(true);

  if (!lox.begin()) while (1);
  Serial.println("UNO READY - BNO055 + VL53L0X");
}
void loop() {
  sensors_event_t event;
  bno.getEvent(&event);
  pitch = event.orientation.x;
  roll  = event.orientation.y;
  rawpitch = pitch;
  rawroll  = roll;
  if (roll < 0) { roll *= -1; roll = 90 - roll; }
  else if (roll > 0) roll = 90 + roll;
  if (pitch > 0 && roll > 0) {
    servo2.write(pitch);
    servo1.write(roll);
  }
  VL53L0X_RangingMeasurementData_t measure;
  lox.rangingTest(&measure, false);
  distance_perp = (measure.RangeStatus != 4) ? measure.RangeMilliMeter : -1;
  if (distance_perp >= 0) {
    distance_real = getRealDistance(distance_perp, rawpitch, rawroll);
    Serial.print("DIST:");
    Serial.println(distance_real, 2);
  }
  delay(50);
}
```

The control loop executes at approximately 20 Hz and updates both servo angles according to the IMU feedback. The distance computation corrects for tilt-induced projection errors using:

$$d_{\text{real}} = \frac{d_{\perp}}{\cos(\text{pitch}) \cdot \cos(\text{roll})}$$

where $d_{\perp}$ is the perpendicular distance measured by the VL53L0X sensor. The corrected distance d_{real} is transmitted over UART to the Teensy controller for integration into the overall height control loop.

7 Demonstration Plan

To validate the stabilization system, a concise demonstration procedure was established to show that the platform maintains horizontal alignment and prevents spillage during different disturbance conditions.

7.1 Setup and Safety

Before testing, the following checks are performed: verification of electrical and mechanical connections, battery voltage (> 14 V per cell), 300 A switch function, emergency stop accessibility, and workspace clearance. A standard cup filled to 80% with colored water serves as the test payload, allowing clear visual confirmation of stability and spillage prevention.

7.2 Demonstration Sequence

Stage 1 – Initialization (2 s): Power on the system and confirm successful startup of all subsystems — Teensy 4.1, Arduino UNO, both IMUs, Moteus controllers via FDCAN, and the VL53L8CX distance sensor. *Success:* All modules initialize within 2 s, no serial errors, platform levels automatically.

Stage 2 – Static Stability (30 s): With the platform stationary, observe that the liquid surface remains level. *Success:* Orientation remains within ±3°, no visible motion, steady-state error = 0.

Stage 3 – Manual Tilting (2 min): Manually tilt the base up to 20° in pitch, roll, and combined directions. Include slow oscillatory movement to simulate walking motion. *Success:* Platform maintains ±3° alignment, reacts within 200–500 ms, and returns level in ¡ 1 s without spillage.

Stage 4 – Combined Disturbances (1 min): Introduce mixed pitch, roll, and height changes, including small impulse disturbances. *Success:* Stable control across all DOFs, minimal liquid perturbation, no saturation or loss of control.

7.3 Expected Outcome

Preliminary results confirm angular stabilization within ±2–3°, fast recovery (1 s), and zero spillage for base tilts up to 20°. The platform responds smoothly and continuously for more than 2 hours per battery cycle, validating the system's performance and overall design goal of maintaining beverage stability during motion.

8 Conclusions

The Hold my Coffee project showcased the active stabilization of a platform holding a beverage by sensing, controlling, and actuating the platform to mitigate the tilting induced by the motion of the base. The stabilized platform was demonstrated to track both the orientation and the height of the base, enabling to carry a full cup over different ground types and through highly dynamic environments.

8.1 Key Achievements

We successfully prototyped a closed-loop multi-axis stabilization system with autonomous actuation. The control hardware was realized by coupling three Moteus motor controllers and two servo motors in a dual microcontroller design [11]. The mechanical hardware was based on 3D-printed ABS and PETG parts, aluminum extrusion structure elements, and precision MGN linear rails, achieving the desired stiffness and modularity.

The implemented software is capable of responsive stabilization with the 20 Hz control update rate, cascaded PID control, and multi-layered rejection of disturbances. We achieved tilt and position measurement using quaternion-based orientation sensing [10][18] to avoid singularities while providing orientation feedback and vertical position control to maintain the platform height using the distance sensor [7].

The power system incorporates multiple safety layers including a 300A waterproof ignition-protected switch, automotive-grade fuses, and low voltage monitoring to ensure safe operation during demonstrations and regular use.

8.2 Challenges Encountered

There were a few open issues and challenges throughout the design process, starting with the motor controller configuration [14]. The lack of documentation and command structure descriptions required some trial-and-error to figure out correct commands and parameter ranges. There were mechanical assembly challenges regarding the alignment of the motor axes and the elimination of backlash in the connecting linkages. Integration of the power system required careful management to ensure that the switching noise from the motors did not introduce measurement noise.

PID gain tuning presented significant challenges due to the varying center of mass as liquid level changes in the cup. The gains needed to provide sufficient disturbance rejection while avoiding oscillation required iterative empirical adjustment across multiple test scenarios.

8.3 Performance Evaluation

The stabilized platform is effective in holding a beverage steady during transport in various situations, such as walking on flat ground, going upstairs and downstairs, and balancing against hand tilting. Qualitative testing shows that the platform can keep the liquid surface horizontally leveled to within a few degrees during typical disturbances. The structured demonstration protocol validates that the system maintains orientation within ± 3 during manual tilting up to 20° and prevents liquid spillage under realistic operating conditions. Battery runtime testing revealed a runtime of around 2 hours during continuous use.

8.4 Potential Improvements

There is a variety of improvement opportunities to further enhance the system for future iterations. State-of-the-art disturbance rejection methods such as ADRC [16][17] could be considered for even better performance. Adaptive control strategies could be explored to achieve consistent stabilization performance when the payload mass changes [12]. A higher resolution sensor with a faster update rate could be considered to increase the bandwidth of the control [18].

Additional sensors such as accelerometers on the base platform could provide feedforward compensation, allowing the system to anticipate disturbances before they affect the upper platform. Integration of wireless monitoring and data logging capabilities would facilitate performance analysis and gain tuning optimization.

8.5 Applications and Future Work

Active stabilization platforms have various applications beyond beverage transport, including medical equipment transport [19], drone delivery platforms [12], and autonomous mobile robots [13][15]. Future work could consider integrating the stabilization platform with an autonomous mobile base or exploring model predictive control approaches [5]. The use of the platform for educational purposes and hands-on learning of control and mechatronics is another potential application.

The system demonstrated that sophisticated stabilization technology could be implemented with off-the-shelf components and open-source tools and made accessible to students and hobbyists to explore and learn more advanced control concepts through experimentation.

References

[1] J. Hilkert, "Inertially stabilized platform technology concepts and principles," *IEEE Control Systems Magazine*, vol. 28, no. 1, pp. 26–46, 2008.

[2] M. K. Masten, "Inertially stabilized platforms for optical imaging systems," *IEEE Control Systems Magazine*, vol. 28, no. 1, pp. 47–64, 2008.

[3] P. J. Kennedy and R. L. Kennedy, "Direct versus indirect line of sight (los) stabilization," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 1, pp. 3–15, 2003.

[4] A. M. Sabatini, "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation," *Sensors*, vol. 11, no. 10, pp. 9182–9206, 2011.

[5] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Princeton University Press, 2021.

[6] Y. Li, K. H. Ang, and G. C. Chong, "Pid control system analysis and design," *IEEE Control Systems Magazine*, vol. 26, no. 1, pp. 32–41, 2006.

[7] H. Rapp, M. Frank, F. A. Hamprecht, and B. Jähne, "Time-of-flight sensors for robotic applications: A comprehensive review," *IEEE Sensors Journal*, vol. 20, no. 12, pp. 6238–6247, 2020.

[8] S. Corrigan, "Introduction to the controller area network (can)," *Texas Instruments Application Report SLOA101*, pp. 1–17, 2002.

[9] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[10] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for imus and margs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[11] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[12] H. Liu, W. Zhao, F. L. Lewis, and S. S. Ge, "Adaptive control for quadrotor uav transporting cable-suspended payload with unknown mass," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2018–2029, 2023.

[13] J.-H. Kim, J.-H. Park, and J.-J. Lee, "Stabilization control for humanoid robot to walk on uneven terrain," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5684–5693, 2020.

[14] Q. Chen, Y. Wang, H. Zhang, and X. Liu, "Design and implementation of a three-axis active stabilized platform based on brushless dc motor," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 11, pp. 9540–9550, 2020.

[15] C. Zhou, X. Wang, Z. Li, and N. Tsagarakis, "Robust control of quadruped robots with active compliance for stable locomotion," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4951–4958, 2021.

[16] F. Wang, R. Wang, E. Liu, and W. Zhang, "Stabilization control method for two-axis inertially stabilized platform based on active disturbance rejection control with noise reduction disturbance observer," *IEEE Access*, vol. 7, pp. 99 521–99 529, 2019.

[17] L. Guo, J. Wang, S. Zhou, and C. Liu, "Disturbance observer-based control for gimbal servo system with multiple disturbances," *IEEE Access*, vol. 9, pp. 21 879–21 888, 2021.

[18] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics*, pp. 1–7, 2011.

[19] W. Zhang, Y. Liu, M. Chen, and H. Wang, "Design and control of an active stabilization system for medical equipment transport," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1923–1932, 2020.