PETER Project Summer Internship Update Report

Universidad Politécnica de Madrid (UPM)
Centro de Automática y Robótica (CAR)
Robótica y Cibernética

Mentor: Jorge Garcia Samartin (PhD Candidate)
Supervisor: Prof. Dr. Antonio Barrientos

Emirhan Yolcu
Middle East Technical University
emirhanyolcu17@gmail.com

15.09.2025

Introduction

This report has been prepared to provide a clear and structured overview of the work conducted on PETER during my internship period in June and July 2025. At the beginning of the project, PETER was in a basic configuration with one IMU and one ToF sensor integrated into a single module (Figure 1). The main objective of my work was initially to implement closed-loop control for this single-module system, and subsequently to advance the design into a modular structure.

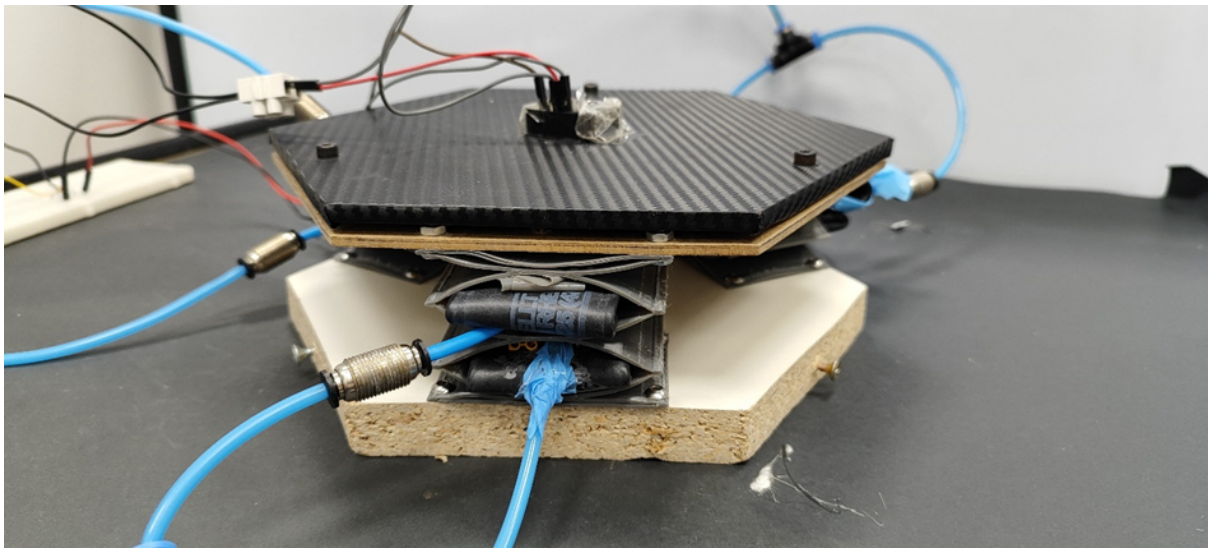

Figure 1: Example photo of single module version of PETER

Single Module Closed Loop Control

At the initial stage of developing the closed-loop control for PETER, fundamental reference definitions were introduced. The height of the system was defined with respect to the ground plane, and the central point was designated as the origin (0,0) corresponding to the condition in which the module's surface is fully parallel to the ground. The control code for the single module setup can be accessed from the GitHub repository under PETER/code_matlab

In the next step, the control scheme illustrated in Figure 2 was developed.

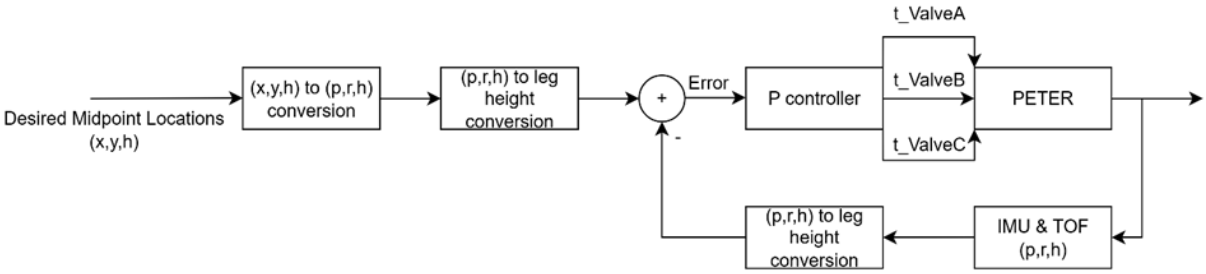

Figure 2: Control loop of single module Peter

The desired position of the platform is first defined in Cartesian coordinates (x, y, h). These coordinates are then converted into the corresponding pitch, roll, and height values (p, r, h) by using the MATLAB function named "xyh_to_prh". In the following stage, the (p, r, h) values are further transformed into the required leg heights by using the MATLAB function named "prh_to_leg_heights". PETER, equipped with IMU and ToF sensors, provides real-time measurements of pitch, roll, and height. These measurements are fed back into the system and transformed again into leg height representations by using "prh_to_leg_heights" function for error evaluation. The loop continues until the desired and measured states converge, ensuring that the platform maintains the commanded position. In addition to the control loop, the measured pitch, roll, and height values from the sensors were converted into Cartesian coordinates (x, y, h) through the MATLAB function "prh_to_xyh". This supplementary step enabled direct comparison of current and desired states, facilitating debugging and precise error monitoring.

—

xyh_to_prh Function – Inverse Kinematic Transformation

Purpose: The `xyh_to_prh` function establishes the inverse kinematic relationship between the desired position of the platform's geometric center and its orientation parameters—pitch (p), roll (r), and height (h). This mapping is essential for the control loop since commands in Cartesian coordinates (x, y, h) must be translated into angular and vertical setpoints that the physical system can achieve.

Inputs: Desired coordinates (x_d, y_d, h_d) **Outputs:** Orientation parameters (p, r, h)

Conceptual Description: In PETER's configuration, the geometric center moves in 3D space as a result of pitch and roll tilts. Each tilt produces a displacement of the center relative to the neutral orientation. Therefore, to achieve a desired center position, one must determine the necessary tilting angles. This is a nonlinear relationship, as the projection of the surface normal on the global frame changes with both pitch and roll.

Mathematical Model: The platform surface normal in its neutral orientation is defined

as:

$$\vec{n_0} = [0, 0, 1]^T$$

After the rotations, the new normal is computed as:

$$\vec{n} = R(p, r)\vec{n_0}$$

where $R(p, r)$ is the combined rotation matrix defined as:

$$R(p, r) = R_x(r)R_y(p) = \begin{bmatrix} \cos r & \sin r \sin p & \sin r \cos p \\ 0 & \cos p & -\sin p \\ -\sin r & \cos r \sin p & \cos r \cos p \end{bmatrix}$$

The geometric center of the platform can thus be expressed as:

$$\vec{c}(p, r, h) = h \cdot \vec{n} = \begin{bmatrix} h \cdot n_x \\ h \cdot n_y \\ h \cdot n_z \end{bmatrix}$$

The goal is to find (p, r, h) such that $\vec{c}(p, r, h)$ matches the desired (x_d, y_d, h_d). To achieve this, a nonlinear cost function is defined:

$$J(p, r, h) = (x_d - c_x)^2 + (y_d - c_y)^2 + (h_d - c_z)^2$$

This function is minimized using MATLAB's `fminsearch`, which iteratively searches for the pitch, roll, and height values that minimize the positional error.

Physical Interpretation: Geometrically, this function determines how much the platform should tilt to shift its center to a specified spatial position while maintaining a certain height. In practical terms, the output (p, r, h) defines the equilibrium orientation that the controller must maintain for the platform's center to coincide with the commanded point in 3D space. This function forms the backbone of the closed-loop control architecture, linking high-level Cartesian targets to low-level actuation commands.

—

prh_to_leg_heights Function – Forward Kinematic Transformation

Purpose: The `prh_to_leg_heights` function performs the forward kinematic mapping from the platform's orientation and center height to the individual leg lengths. It translates the global pose parameters (p, r, h) into the actuator-level quantities required to realize that pose physically.

Inputs: Orientation (p, r, h) **Outputs:** Leg heights (h_A, h_B, h_C)

Platform Geometry: The platform is modeled as a rigid equilateral triangle with side radius L. In its neutral state, the corner points are defined with respect to the center as:

$$\vec{A_0} = [L, 0, 0]^T, \quad \vec{B_0} = [-L/2, \sqrt{3}L/2, 0]^T, \quad \vec{C_0} = [-L/2, -\sqrt{3}L/2, 0]^T$$

The vertical position of each corner is determined by applying the desired pitch and roll rotations to these reference vectors.

Mathematical Derivation: The orientation of the platform is expressed through the rotation matrix $R(p, r)$. Each corner vector is rotated and translated according to the desired center height:

$$\vec{A_1} = R(p, r)\vec{A_0} + [0, 0, h]^T$$

$$\vec{B_1} = R(p, r)\vec{B_0} + [0, 0, h]^T$$

$$\vec{C_1} = R(p, r)\vec{C_0} + [0, 0, h]^T$$

Since the ground plane is at $z = 0$, the leg extensions are simply the z-components of these rotated corner positions:

$$h_A = (\vec{A_1})_z, \quad h_B = (\vec{B_1})_z, \quad h_C = (\vec{C_1})_z$$

In implementation, the function uses Rodrigues' rotation formula for computational efficiency:

$$\vec{v_{rot}} = \vec{v}\cos\theta + (\vec{k} \times \vec{v})\sin\theta + \vec{k}(\vec{k} \cdot \vec{v})(1 - \cos\theta)$$

where $\vec{k}$ is the rotation axis derived from pitch and roll directions, and θ is the combined tilt magnitude.

Physical Interpretation: This function establishes how each actuator must adjust its length to create the commanded tilt and height. When the platform tilts forward, one leg extends while another retracts, maintaining the planar surface at the specified inclination. Accurate implementation of this function ensures smooth and coordinated motion across all actuators, minimizing mechanical stress and maintaining geometric stability during operation.

—

prh_to_xyh Function – Cartesian Mapping and Feedback Validation

Purpose: The `prh_to_xyh` function is used primarily for feedback processing. It converts the measured or simulated pitch, roll, and height values back into Cartesian coordinates, allowing the controller to evaluate the spatial deviation between desired and actual positions.

Inputs: Orientation (p, r, h) **Outputs:** Cartesian position (x, y, h_{out})

Mathematical Model: The local surface normal of the platform after rotation is:

$$\vec{n'} = R(p, r)\vec{n_0}$$

where $\vec{n_0} = [0, 0, 1]^T$. By scaling this normal vector by the current height h, the new geometric center coordinates are derived as:

$$x = h \cdot n'_y, \quad y = h \cdot n'_x, \quad h_{out} = h \cdot n'_z$$

These relations describe how the tilting of the platform projects its center in the global coordinate system.

Implementation Note: This function plays a critical role in closed-loop validation. The measured IMU (pitch, roll) and ToF (height) readings are transformed into Cartesian coordinates using this mapping, enabling direct comparison with the target (x_d, y_d, h_d). It thus provides an intuitive error metric in the same domain as the command signal, simplifying controller tuning and debugging.

Physical Interpretation: Conceptually, this function performs the inverse of `xyh_to_prh`. It quantifies how the actual orientation affects the spatial displacement of the platform's center. Accurate calculation of this mapping is crucial for evaluating control accuracy, analyzing steady-state error, and diagnosing sensor misalignments or actuator calibration issues.

—

Controller

In this process, mathematically modeling PETER proved to be very challenging, so instead of relying on exact calculations for the control parameters, we applied tuning methods. After initially implementing a basic P controller, we also tested a PI controller. However, due to the high level of noise in the sensor measurements, the integral action could not operate effectively, which limited the performance of the PI controller. After

comparing the desired and current leg lengths, the P controller determines a positive or negative valve opening duration for each leg based on the corresponding error value. These valve commands are then transmitted serially to the Arduino for actuation.

Two Module PETER

For the two-module version of PETER, the control system does not significantly change, but it requires the use of two ToF sensors and two IMU sensors. Connecting multiple sensors introduces the issue of I^2C address conflicts. To resolve this, one of the IMUs was connected directly to the 5V line to separate its addressing, while the ToF sensors were assigned new addresses in software by connecting them to some digital pins. All the necessary Arduino code for this setup can be found in the GitHub repository under peter_arduino/Peter 4 Sensors. Uploading this code directly to the Arduino is sufficient to handle both valve control and sensor data acquisition. Similar to the single-module case, the legs are first considered as a whole, and the required lengths A, B, and C are calculated according to the desired (x, y, h). These lengths are then divided by two so that both modules share the same leg extension. The resulting values are sent to two separate controllers, each responsible for its own module. With their dedicated sensors, the two modules independently regulate their leg lengths, enabling the platform to reach the desired position successfully. The control code for the two-module setup can be accessed from the GitHub repository under PETER/code_matlab/2 module.

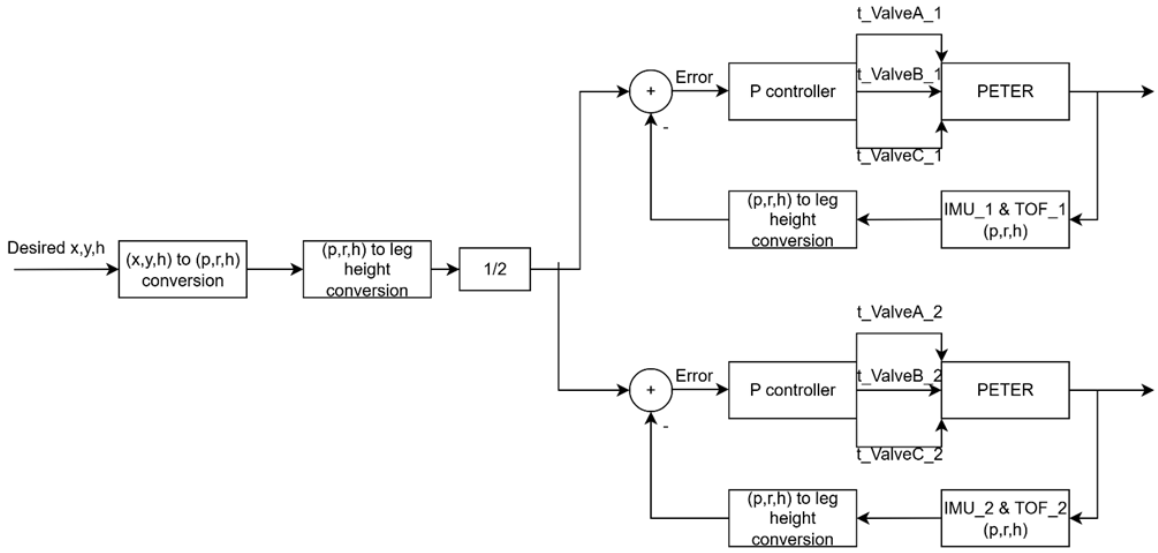

Figure 3: Control loop of two module Peter

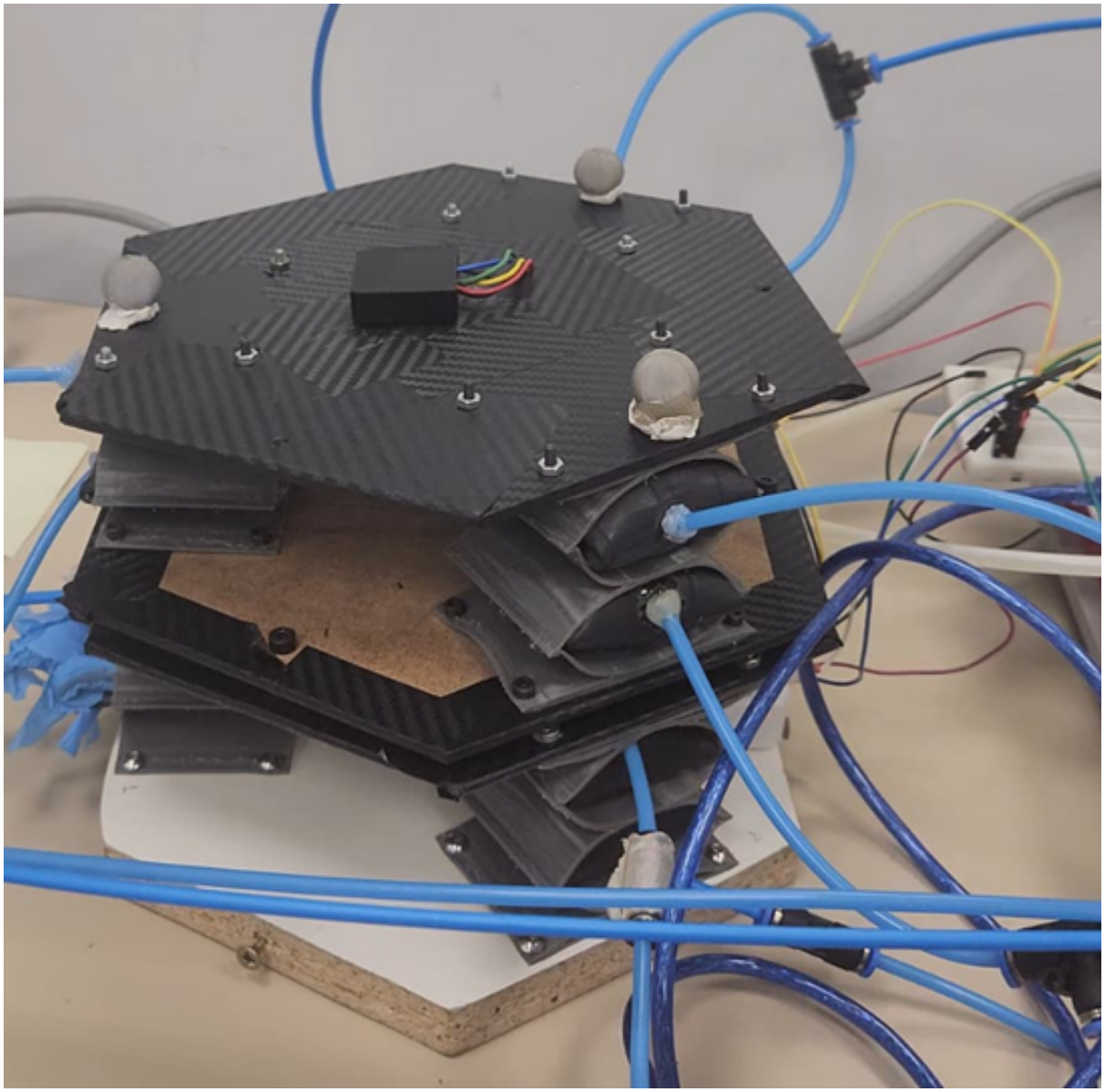

Figure 4: Photo of two module Peter

Test & Validation

The testing process of PETER was carried out in two stages. In the first stage, the desired point was set to (0,0), and disturbance rejection capabilities were evaluated. A video of this test can be accessed through the link provided below. `https://youtube.com/shorts/rUIRMNMMSJA?feature=share`

The second stage was conducted in a more scientific setup using an environment equipped with OptiTrack cameras. In this setup, both the rotational and positional data of PETER were recorded. The rotational information obtained from the OptiTrack system was then used to calculate the platform's center position with our own algorithm, and these results were compared against the ground-truth position measurements from OptiTrack.

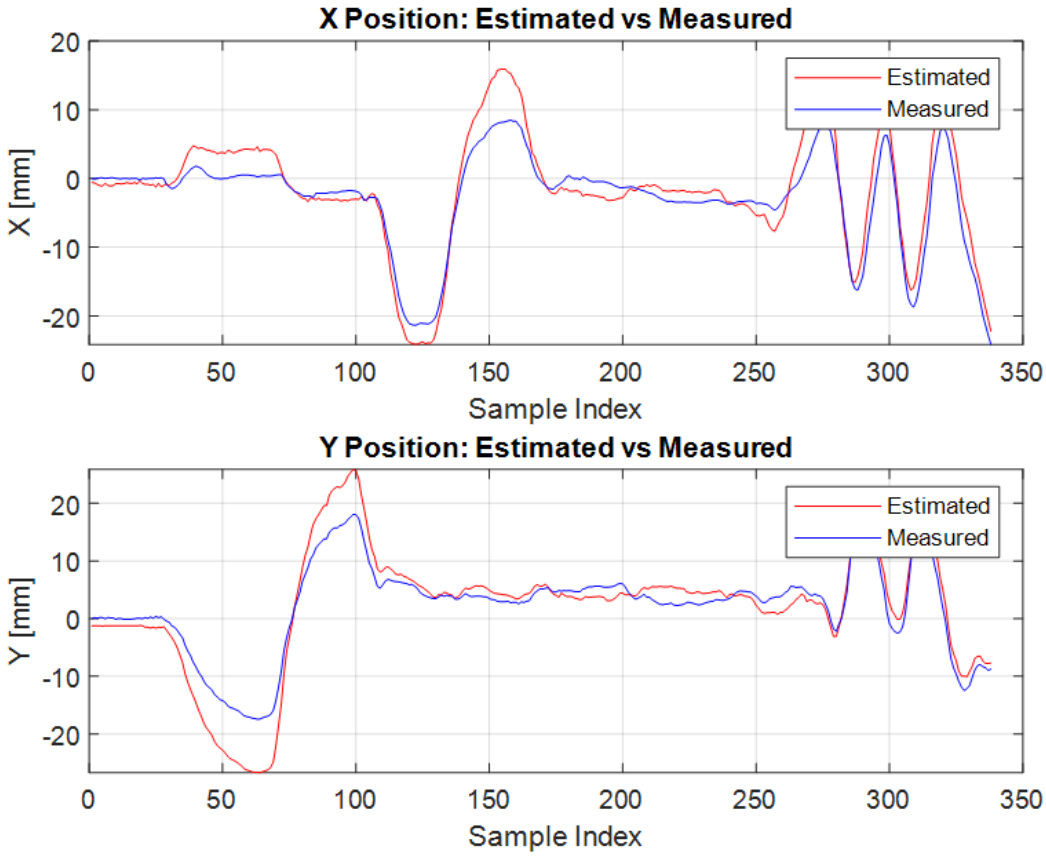

Figure 5: OptiTrack Test Results

The required test data and the codes used to acquire data from the OptiTrack system can be found in the GitHub repository under code_matlab/Validation.

Future Work

Although progress has been made in the PETER project, it is not yet completed and remains an open-ended system with significant potential for future improvements. One of the most important developments will be the experimental modeling of PETER. By applying different input signals to the system and recording the corresponding rotational outputs with OptiTrack, an experimental model can be derived using MATLAB's system identification tools. In addition, alternative control methods such as PD or LQR can be tested to achieve more optimal results compared to the current approach. While IMU measurements have proven to be relatively stable, the ToF sensor readings have shown considerable instability and noise. Therefore, exploring alternative distance measurement techniques could substantially improve accuracy and robustness in the control of PETER. As a final recommendation, since address conflicts are a common issue in I^2C communication, the use of an external multiplexer (mux) would be highly beneficial. Implementing

a mux can greatly simplify cabling and resolve many potential communication problems, thereby improving the overall reliability of the system.