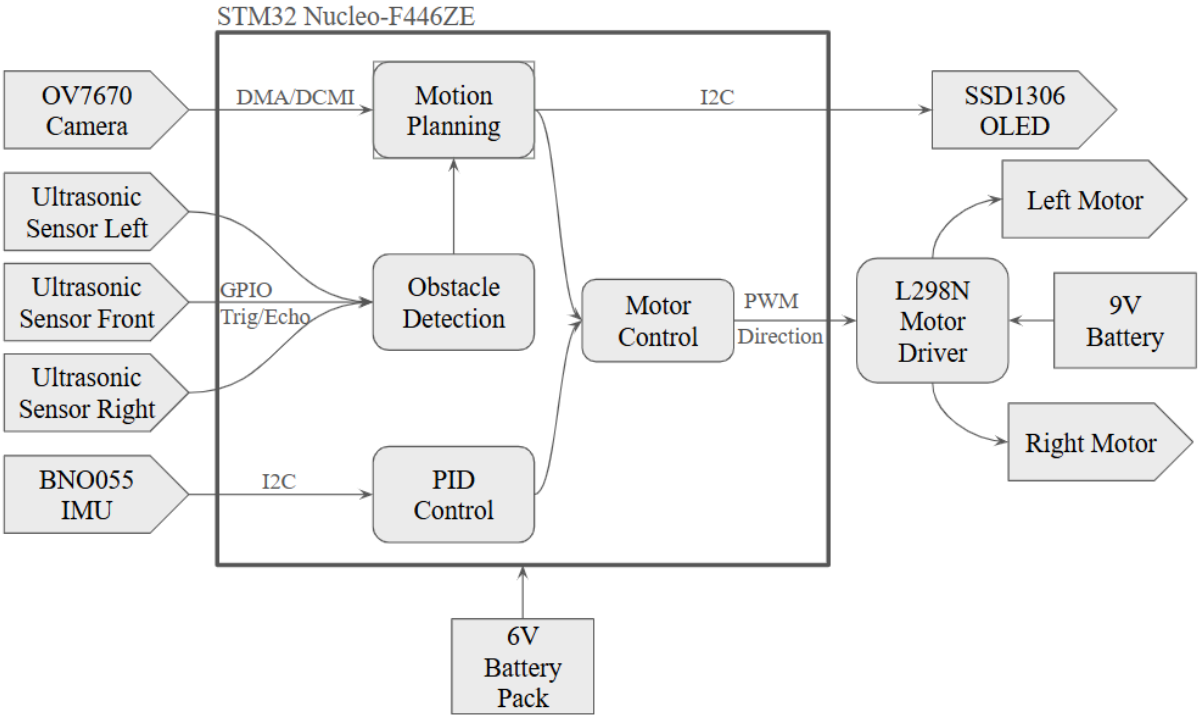
Ethan Lam 1010153389
Madeline Zhao 1010004175

Section 1 — Project Overview & Analysis

We demonstrated an STM32-based robot capable of self-balancing for up to 10 seconds, navigating toward a red target, and detecting obstacles in its path.



Section 2 — Bandwidth / Throughput Analysis

Component	Limiting Parameter	Analysis
OV7670 DMA/DCMI	Frame transfer rate / processed frame rate	Each frame contained 144 × 174 × 2 = 50,112 bytes of image data. With the camera configured for about 15 fps, the overall theoretical image data rate is about 50,112 × 15 = 751,680 bytes/s = 0.75 MB/s. In practice, the effective frame rate was likely lower because after the DMA completed, the CPU still had to run colour tracking and motion logic. Thus, the system stayed below the theoretical maximum because processing time limited how quickly frames could be turned into decisions.
SSD1306 I2C	I2C bus speed / transfer overhead	At 100kHz, the theoretical bandwidth was 12.5kB/s, so a full-screen refresh could only reach about 9-10 refreshes/s. In practice, the OLED ran far below that because the code only updated a 64 × 8 region, only

		when the text changed, and only every 300 ms at most. So the actual update rate was far below the theoretical by design, since the display was deliberately throttled to reduce I2C traffic and overall CPU overhead to allow for work on more critical tasks.
HC-SR04	Measurement update rate / blocking read latency	Each read used a 10 µs trigger pulse and up to a 12 ms timeout, so one sensor could theoretically support about 83 readings/s. In practice, only one of the three sensors was polled every 100 ms, so each sensor updated once every 300 ms or about 3.3 Hz. This was much lower than theoretical because the reads were blocking and were intentionally staggered, so sensing would not occupy too much time and interfere with balancing.
BNO055	IMU sampling rate	The controller has a maximum theoretical sample rate of 100 Hz, or a minimum sample period of 10 ms. In practice, the rate at which IMU data was sampled was slightly slower, measured to be 15-50 ms depending on which other processes occurred in the loop, and less consistent due to the use of polling instead of interrupts and irregular timing of other tasks such as updating the screen.
DC Motors	Input / Output power	The available input power was insufficient for the motors to maintain a balanced state. See Appendix A for details.

Main bottleneck:
The main bottleneck was the motor/power subsystem, because the available power and torque were insufficient for the motors to make fast enough corrective movements to maintain a stable and balanced state. The HC-SR04 obstacle-sensing path was still a significant software timing bottleneck, but it was not the dominant factor when it came to limiting the robot's overall performance.

Single change that would most improve performance:
Use high-torque motors and a better power source/power-delivery design so the robot can apply quicker and stronger corrective movements during balancing to help it stabilize and reduce drift when moving.

Ethan Lam 1010153389
Madeline Zhao 1010004175

Section 3 — Code Contribution Table

Module	Source Type	Notes
OV7670 API	Modified from existing source	Adapted from ECE342 Lab 5 camera code [1]
SSD1306 API	Modified from existing source	Core display/I2C logic from Lab 4; font table generated with Gemini and manually verified [2, 3]
HC-SR04 API	Written from scratch	From HC-SR04 datasheet timing, NXP Application Note on DWT cycle counting, and lecture on GPIO for pin mapping [4, 5, 6]
BNO055 API	Written from scratch	Based on information from BNO055 datasheet [7]
PID Controller	Written from scratch	From prior knowledge (ECE311)
Color Detection	Written from scratch	Informed by online research and manual calibration for custom HSV thresholding and binning [8]
Motion Planning	Written from scratch	From careful, manual calibration for custom camera/ultrasonic safety logic

Section 4 — Project Reflection

The original goal of this project was to create a self-balancing robot capable of carrying and transporting items. The planned system used IMU data and a continuous PID controller to maintain balance, a camera to determine the relative direction of a target for steering, an ultrasonic sensor to suppress forward motion when obstacles were too close, and an OLED display to show system status.

We did not fully achieve the original goal of our project due to its complexity. Instead, we built a proof-of-concept robot that could self-balance for up to about 10 seconds, navigate toward a red target, and detect obstacles in its path. The main reason we reduced the scope was that combining stable balance control with real-time sensing, steering, and actuation on the same physical frame proved to be unreliable at times and challenging given the hardware we were working with. This was consistent with our proposal, which stated that stable self-balancing under varying conditions would be the primary technical challenge of this project.

Ethan Lam 1010153389
Madeline Zhao 1010004175

If we were to repeat this project, one specific technical decision we would change would be a more comprehensive and informed redesign of the robot's frame and mechanical architecture. We would integrate higher-torque motors with encoders to allow for more precise feedback and stabilize the power supplied to the board and motor driver. We would also improve the robot's weight distribution by strategically raising the center of gravity to treat it more like an inverted pendulum, whereby its higher mass distribution would increase the moment of inertia, slowing how quickly it tips and thus, providing a larger window for the control loop to execute corrective movements. These changes would likely enhance the robot's balance stability and minimize drift, making the control loop significantly more reliable than further software tuning alone.

One important privacy issue is that the camera can capture nearby people or parts of the surrounding environment, while a related safety and ethical issue is that a moving self-balancing robot could behave unpredictably if its sensing becomes unreliable. We addressed the privacy issue by processing camera frames locally on the STM32 for immediate target tracking only, rather than storing, transmitting, or using them for broader analysis. We addressed the safety issue mostly in software, whereby the robot defaulted to a "hold" state when no target was found, stopped motor output once the tilt deviated by more than 45° from the calibrated angle, blocked forward or turning motions when ultrasonic thresholds detected nearby obstacles, and did not attempt normal operation if the IMU failed to initialize. Overall, these choices helped to reduce unnecessary visual data collection of its environment and lowered the risk of unsafe, unpredictable motion by employing limited, local processing and a very conservative fallback behavior.

Section 5 — Workload Split

Ethan was mainly responsible for the robot frame construction, ultrasonic obstacle-detection logic, motion-planning logic, and firmware debugging/optimization. This involved linking the camera and ultrasonic data to the robot's steering logic while refining its tracking and safety behavior.

Madeline was mainly responsible for the balance controller design and tuning, motor-control implementation, system integration, and hardware testing/debugging. This included integrating IMU feedback with PID-based balance control and validating the robot's behavior on hardware.

Ethan Lam 1010153389
Madeline Zhao 1010004175

References

[1] K. Ganesan, "Lab 5: DMA/DCMI using the OV7670 camera," ECE342H1 S Course Lab Handout, Dept. Elect. & Comp. Eng., Univ. of Toronto, Toronto, ON, 2026. [Online]. Available: https://q.utoronto.ca/courses/419379/pages/lab-5a-and-5b?module_item_id=7591710. [Accessed Apr. 6, 2026].

[2] K. Ganesan, "Lab 4: I2C peripheral," ECE342H1 S Course Lab Handout, Dept. Elect. & Comp. Eng., Univ. of Toronto, Toronto, ON, 2026. [Online]. Available: https://q.utoronto.ca/courses/419379/pages/lab-4?module_item_id=7570477. [Accessed Apr. 6, 2026].

[3] Google Gemini, "5x7 ASCII font table in C-style hex array," Mar. 24, 2026. [Large Language Model]. [Online]. Available: google.com. [Accessed Apr. 6, 2026].

[4] SparkFun Electronics, "Ultrasonic Ranging Module HC - SR04," SparkFun Electronics, Boulder, CO, 2026. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Accessed Apr. 6, 2026].

[5] NXP Semiconductors, "AN14211: How to Utilize Trace Components on i.MX RT1180," NXP Documentation, Dec. 10, 2024. [Online]. Available: https://docs.nxp.com/bundle/AN14211/page/topics/using_dwt_cycle_counter.html. [Accessed Apr. 6, 2026].

[6] P. Naseri, "Module 1: GPIO 1," ECE342H1 S Course Lecture Slides, Dept. Elect. & Comp. Eng., Univ. of Toronto, Toronto, ON, 2026. [Online]. Available: https://q.utoronto.ca/courses/419379/pages/module-1-on-chip-peripherals-lectures-2-6?module_item_id=7543115. [Accessed Apr. 6, 2026].

[7] Bosch Sensortec, "BNO055 Intelligent 9-axis Absolute Orientation Sensor Data Sheet," Bosch Sensortec Media, Oct. 2021. [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf. [Accessed Apr 6, 2026]

[8] Wikipedia contributors, "YCbCr: JPEG conversion" and "HSL and HSV: Formal derivation," Wikipedia, The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/YCbCr and https://en.wikipedia.org/wiki/HSL_and_HSV. [Accessed Apr. 6, 2026].

[9] Adafruit, "DC Gearbox Motor - 'TT Motor' - 200RPM - 3 to 6VDC" Adafruit Industries, [Online]. Available: https://www.adafruit.com/product/3777 [Accessed Apr 6, 2026]

Ethan Lam 1010153389
Madeline Zhao 1010004175

Appendices

Appendix A: Motor Power

Each motor has a maximum input voltage of 6V and a stall current of 1.5A, but they are powered together by a 9V alkaline battery, which can deliver a maximum current of approximately 400mA. The stall torque (maximum torque) of each motor is 0.8 kgcm, or approximately 0.0785Nm, so, at 200mA, the stall torque is estimated to be 0.013 Nm [9].

Assuming an average coefficient of rolling resistance of 0.02 for rubber wheels, a diameter of 65mm, and that the weight is distributed evenly, the rolling resistance per wheel is $Fr = 0.02 * (0.5kg * 9.81m/s^2) = 0.0981N$, and the torque needed to overcome this is $\tau = Fr * r = 0.0981N * 0.0325m = 0.00319Nm$.

The maximum power output of a DC motor occurs at around 50% stall torque and 50% max RPM. In the case of our robot, this would mean 0.00652 Nm and 125RPM. Meanwhile, at maximum input power of 6V and 1.5A, the motor should be able to provide 0.03925 Nm of torque at the same RPM.

The rolling resistance consumes almost half of the maximum torque (0.00319/0.00652), making it difficult for the wheels to quickly adjust the robot's position since there is less available power.