Enhancing the Wombat: A Cost-Effective RaspberryPi5 Upgrade and Firmware Extensions for the Botball Controller

Tobias Madlberger[1], Leander Klik, Matthias Greil
HTL St. Pölten, Austria
[1]Corresponding author: tobias.madlberger@gmail.com

***Abstract*—The Wombat robotics controller, based on the Raspberry Pi 3B+, has been the foundation of Botball's embedded platform since 2020. However, rising demands for real-time computer vision have begun to outpace its capabilities. We present a low-cost, minimally invasive upgrade to the Raspberry Pi 5, paired with firmware extensions that expand functionality and performance. Our approach delivers a 5–13× reduction in inference latency across common CNN workloads, increased performance of the motors and introduces software-controlled servo shutdown. The hardware swap retains compatibility with existing extension boards, costs under $50 net, and is feasible for mass adoption by the 2026 season. Rather than widening the gap, this upgrade democratizes advanced vision pipelines and makes real-time perception accessible to all Botball teams.**

***Index Terms*—Botball, Wombat Controller, Raspberry Pi 5, Motor PWM, Servo Disable, Embedded Robotics**

I. Introduction

The Wombat platform successfully abstracts sensors, actuators, and vision into a student-friendly package [1], [18]. Nevertheless, challenges such as 1-minute boot times and CPU-bound TensorFlow Lite pipelines increasingly limit competitive teams. Rather than designing a new controller from scratch, we explore whether the Raspberry Pi5 released in late 2023 can be retrofitted with minimal hardware churn while unlocking contemporary performance [2].

Our contribution is three-fold:

1) A *drop-in* hardware swap requiring a $7 Micro-HDMI dongle.
2) Firmware updates that (i) improve the motor performance and (ii) introduce a way to disable servos fully.

II. System Overview

A. Hardware Swap: RaspberryPi5

The Pi5 brings a 2.4GHz Cortex-A76 quad-core SoC and up to 8GB LPDDR4X RAM, delivering roughly triple the SPEC int score of the 3B+ [3]. Crucially, the 40-pin GPIO header is electrically identical [4], ensuring compatibility with the Wombat's extension board. Physical differences are limited to dual Micro-HDMI connectors and deeper USB sockets (8mm overhang), neither of which interferes with the stock plastic enclosure.

B. Alternative Low-Cost Vision Controllers

While the Raspberry Pi 5 represents a balanced upgrade path for Wombat, two other budget-oriented AI boards are commonly pitched to K–12 robotics teams:

- **OpenMV Cam H7**: $65 microcontroller board (480 MHz Cortex-M7, 256KB RAM). Draws 110mA idle and manages ∼0.6 FPS on a YOLO demo, making it great for line following but too slow for real-time CNN work. [12]–[14]
- **NVIDIA Jetson Nano (4GB)**: $99 launch MSRP (often $150+ in 2025) with a 128-core Maxwell GPU (roughly 0.5 TFLOPS FP16). Runs YOLOv5s at 6–14 FPS but needs an active cooler and 5–10W power budget. [15]–[17]

TABLE I
Cost/Performance Snapshot (June 2025)

Board	Street Price	YOLOv5s FPS	$ per FPS
OpenMV H7	$65	0.6	$108
Jetson Nano	$149	11	$13.5
Pi 5 (Wombat)	$47*	4.1	$11.5

**Net incremental cost after reselling the retired Pi 3B+.*

Table I shows that the Pi 5 retrofit matches the Jetson Nano in $-per-FPS efficiency within a $2 margin, while offering full pin compatibility with the Wombat stack and consuming half the power. The OpenMV Cam, being an order of magnitude less capable for vision tasks, is better suited as a sensor-node companion rather than a standalone controller.

Takeaway: swapping to Pi 5 keeps total system cost low, matches Nano-class efficiency, and avoids the BOM bloat of a discrete carrier board.

The replacement process is straightforward: open the case, unplug the existing HDMI adapter, and gently remove the Raspberry Pi 3B+ from the GPIO header. As with any electronics handling, care should be taken to avoid static discharge and to prevent bending GPIO pins during insertion. Then, insert the Raspberry Pi 5, connect the new Micro-HDMI adapter, and the upgrade is complete.

C. Firmware Extensions

Motor-Driver Duty-Cycle Fix reduces a 1 ms dead-band that forced a maximum duty cycle of 75%. Shortening the back-EMF sampling window to 104µs mirrors best practices in high-fidelity DC drives [5].

Servo Shutoff in Software The Wombat hardware provides the possibilities to shutoff (not only disable) the servos, which isn't accessible for teams to use in software. This feature can provide more flexibility in certain situations like the setup of robots.

III. Cost and Feasibility

TABLE II
INCREMENTAL BILL OF MATERIALS (MAY 2025)

Component	Qty	Unit Cost (USD)	Sub-Total (USD)
Raspberry Pi 5 (4 GB)	1	60	60
Micro-HDMI Adapter	1	7	7
Total			67
Credit: Retired Pi 3B+			–20 to –30
Net Increment			**37–47**

Looking at Table II, the hardware upgrade requires a relatively minor investment. The most expensive component is the Raspberry Pi 5 itself, priced at $60. Minor accessories such the Micro-HDMI adapter bring the total to approximately $67.

However, one can potentially offset the cost by repurposing or reselling the now-retired Raspberry Pi 3B+, reducing the net upgrade cost to around $39–$49. [1]

IV. Benchmark Results

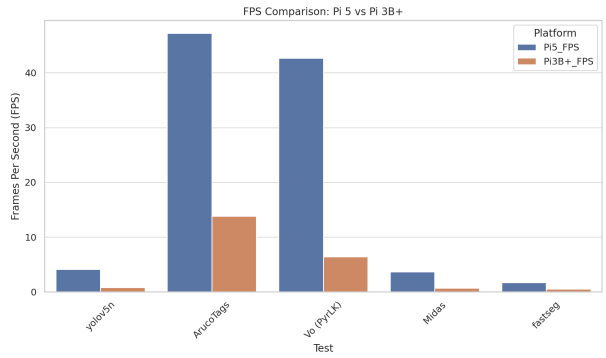

Fig. 1. Inference Latency Comparison of CNN Models on Raspberry Pi 3B+ vs Pi 5.

The benchmarking results clearly demonstrate a dramatic performance uplift with the Raspberry Pi 5 as it can be seen in Table III and Figure 1. All metrics were averaged over 100 inference runs per task using the 4GB model of the Pi 5. The device was mounted inside the Wombat enclosure as described, with an additional CPU fan installed. No thermal throttling or power-saving modes were observed throughout the tests. Across all tasks, inference times were significantly reduced. Most notably:

- **YOLOv5n**: Average latency dropped from 1201ms to 241ms (5× improvement).
- **Visual Odometry (PyrLK)**: From 156ms to just 23ms, enabling real-time motion tracking.
- **Midas Depth Estimation**: Previously impractical at 1.5s, now down to 271ms.

These improvements translate into real FPS gains (e.g., April tag detection now runs at 47Hz), enabling reliable perception pipelines on embedded platforms.

These benchmarks have been achieved with minimal effort and limited optimization, relying solely on standard usage patterns provided by existing inference frameworks - primarily ONNX Runtime for PyTorch - exported models (e.g., YOLOv5n, FastSeg) and TensorFlow Lite for pre-trained models like Midas. For AprilTag detection OpenCV contrib has been used. As such, there remains substantial room for performance improvement, especially for tasks with higher compute demands or non-optimized pipelines.

TABLE III
PERFORMANCE METRICS ACROSS VISION TASKS ON RASPBERRY PI 3B+ AND PI 5.

Metric	Yolov5n	ArucoTags	VO (PyrLK)	Midas	FastSeg
Raspberry Pi 5					
Avg (ms)	241.45	21.18	23.44	271.09	595.27
Min (ms)	233.14	20.75	23.38	270.06	590.02
Max (ms)	266.28	21.67	23.53	271.75	599.23
FPS	4.14	47.21	42.66	3.68	1.68
Raspberry Pi 3B+					
Avg (ms)	1201.24	72.40	155.59	1484.48	1919.29
Min (ms)	1161.91	72.04	155.12	1380.85	1891.69
Max (ms)	1425.45	73.28	157.16	1942.43	2158.21
FPS	0.83	13.81	6.43	0.67	0.52
X-Factor	5.0	3.4	6.6	5.5	3.2

V. Motor Driver Analysis

A. What is BEMF and how is it measured on the Wombat

Back ElectroMotive Force (BEMF) is the voltage generated by a motor as it spins. This self-induced voltage is a direct result of the motor's rotation and can be used to estimate the motor's speed [5]. On the Wombat platform, BEMF is sampled to enable sensorless feedback for closed-loop control. Each motor has a dedicated passive resistor-capacitor network that taps into the motor terminals (MOTxA and MOTxB) and attenuates the signal for safe measurement. This network, shown in Fig. 2, feeds the differential signal into two analog channels of the STM32 coprocessor. A capacitor between the differential lines provides basic low-pass filtering to reduce switching noise from PWM signals.

This circuit is downstream from the Wombat's 6V battery supply voltage, which supplies power to the motors. It's

[1]Estimated resale value based on typical secondhand pricing on electronics marketplaces.

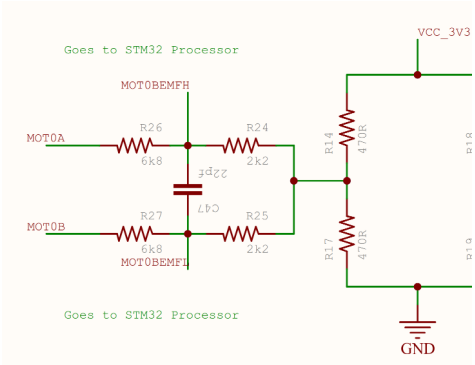

Fig. 2. Motor BEMF sensing circuit

important to note that while the motors are driven from the 6V batteries, the BEMF taps only monitor passive voltages and do not interfere with the drive circuitry.

B. Analysis of KIPR's Implementation

The BEMF measurement and interpretation are performed entirely by the STM32F427 coprocessor. The built-in ADCs sample the attenuated BEMF signals at regular intervals. These raw readings are filtered digitally via a low-pass filter to smooth out PWM-related artifacts. The resulting filtered signal is accumulated over time to produce a value proportional to the motor's relative rotation. This accumulated value is periodically sent to the Raspberry Pi via SPI.

In Libwallaby, this value is exposed as *motor ticks*. Internally, the accumulated BEMF sum is scaled by a predefined factor to approximate encoder ticks. Oscilloscope traces show that new BEMF measurements are taken approximately every 3.6ms.

Fig. 3. Measurement: motor voltage at 50% duty cycle

In the Wombat firmware [10], which runs on the coprocessor, BEMF measurements are performed every fourth iteration of the main control loop. Each iteration of this loop takes approximately $900\mu s$ to execute, resulting in a maximum effective motor "on time" of roughly 75%.

Ideally, the dead-band - during which no PWM is applied to the motor - would only need to last as long as the ADC sampling time. However, due to the motor's inductance, turning off the motor causes a brief negative voltage spike. To avoid corrupting the measurement, a delay must be inserted after disabling the motor before sampling the BEMF voltage.

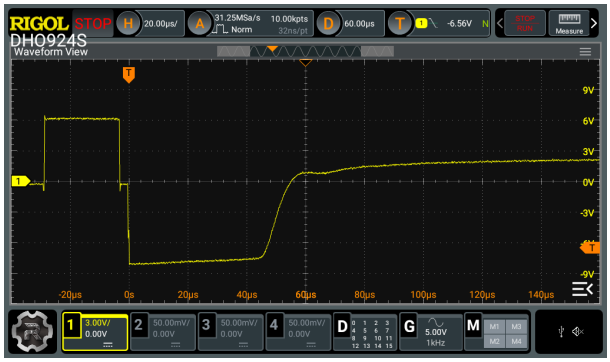

Fig. 4. Measurement: negative voltage spice at the start of BEMF

From figure 5 it can be seen that it takes approximately $80\mu s$ before the voltage has stabilized. It should also be noted that the duration of the voltage spike varies slightly by every measurement. The voltage is stable consistently after $80\mu s$.

C. Improved Solution for Optimal Performance

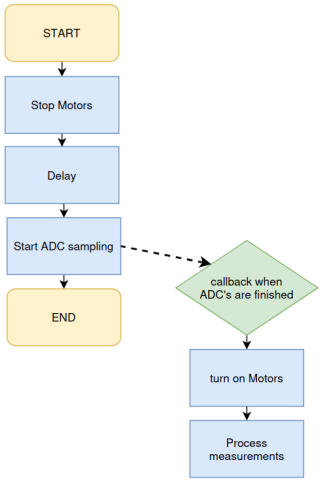

Fig. 5. flowchart - optimized BEMF Implementation

To optimize the BEMF measurement process and reduce CPU load, the new implementation offloads as much work as possible to asynchronous hardware components. Specifically, one of the two DMA controllers on the Coprocessor is now used to handle ADC sampling.

The CPU only initiates the ADC conversion; once the sampling is complete, a DMA interrupt is triggered. This allows the CPU to remain free for other tasks during the sampling period.

When integrating this new implementation into the Wombat-Firmware [10], most of the existing structure can be retained. As in the previous version, BEMF measurement is initiated every fourth iteration of the `main-loop`. The key difference

is that the ADC sampling now occurs asynchronously, and both the processing of the data and re-enabling of the motors are performed in the DMA callback.

A more optimized variant of this approach uses a hardware timer to periodically initiate BEMF measurements. This version was used for the performance measurements described below. Instead of blocking the CPU with a manual delay after turning off the motors, the timer is configured to provide precise microsecond-level timing.

Every 4 ms, the motors are disabled. After a delay of 80 μs - which allows voltage transients to settle - the ADC conversion is triggered.

```
#define BEMF_SAMPLING_INTERVAL 4000 // mu s
#define BEMF_CONVERSION_START_DELAY_TIME 80 // mu s

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    // Timer 6 triggers this callback every microsecond
    if (htim->Instance == TIM6)
    {
        microSeconds++;

        static uint32_t bemfLastStart = 0;
        doEveryXuSeconds(&stopMotors(), BEMF_SAMPLING_INTERVAL, bemfLastStart);
        doAfterXuSeconds(&startBEMFadcConversion(), BEMF_CONVERSION_START_DELAY_TIME, bemfLastStart);
    }
}

// Called by interrupt when ADC2 conversion is complete
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    if (hadc->Instance == ADC2)
    {
        update_motor_cmd();
        processBEMF();
    }
}
```

Listing 1. Improved BEMF measurement using DMA and timer callbacks

D. Measurement with the new BEMF measurement

When comparing the measurements of the optimized version in figure 6 and figure 7 with the legacy version in figure 3 a 770μs decrease of the dead-band can be seen. This equals a reduction of approximately 88%.

E. Performance Comparison

To evaluate the impact of the improved BEMF measurement routine, a performance benchmark was conducted using identical hardware and power conditions. The time required for one full motor rotation was measured for both the legacy and updated firmware implementations:

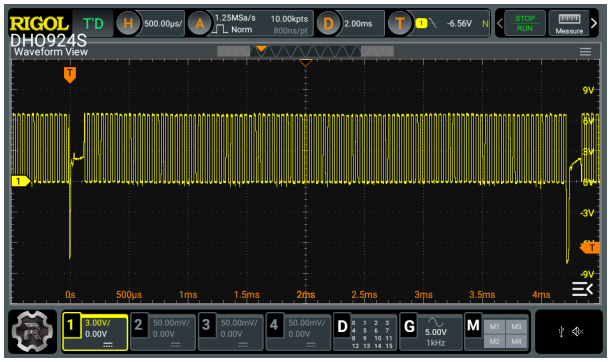

Fig. 6. Measurement: optimized implementation - motor voltage at 50% duty cycle

Fig. 7. Measurement: BEMF measurement cycle with optimized implementation

- **Legacy Implementation:** 1.14s per full rotation
- **Optimized Implementation:** 1.04s per full rotation

This corresponds to an **8.9% improvement** in rotational speed. Over continuous operation, this optimization translates into approximately 11s of time savings per 120s of active movement - a significant gain within the constraints of Botball's competition rules.

The enhanced performance is primarily due to reduced dead time in the motor control loop, achieved by refining the timing of high-impedance phases and optimizing the BEMF sampling window.

VI. Servo Mode

Servos in the Wombat ecosystem can be disabled from moving, but still stop external forces form changing the servo position. In most cases this is wanted, but in come cases like in the setup this can be cumbersome.

Wombats have the hardware to disable the voltage supply of the servos, this allows the free movement of it. The 6V Regulator located on the Wombats is only used for the servo voltage supply. So by disabling the voltage supply it turns the servo off.

To implement the low-level-software for this the STM32 has to toggle the corresponding port pin low which forces the enable pin fo the 6V Regulator low. To avoid damaging

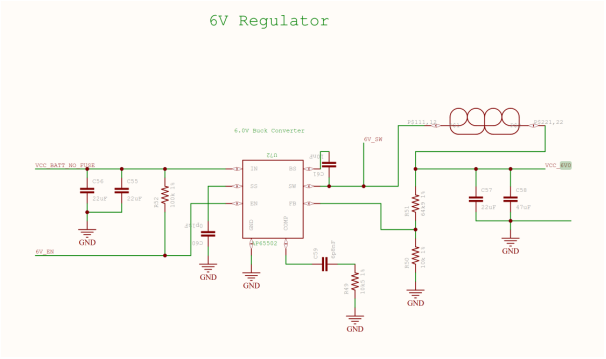

Fig. 8. wombat schematic: 6V regulator [11]

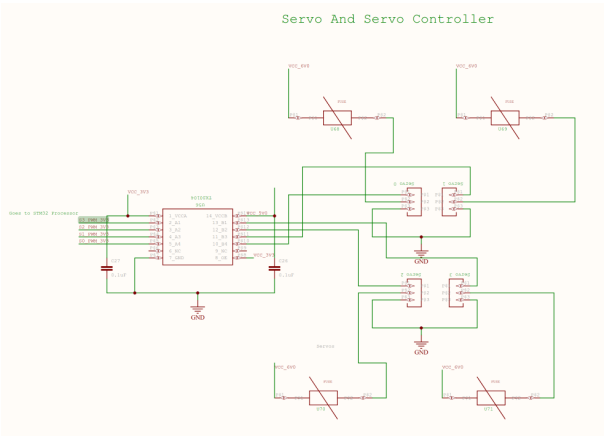

Fig. 9. wombat schematic: Servo Controller [11]

the control circuit in the Servo itself the PWM signal that controls the position should also be turned off.

VII. New Capabilities

Prior to this upgrade, many advanced computer vision pipelines - such as object detection, marker tracking, or depth estimation -were theoretically possible but practically unusable on the Wombat's Raspberry Pi 3B+ due to severe inference latency.

With the Pi 5, these tasks cross into a viable speed bracket:

- **YOLOv5n** inference now runs at over 4FPS, enabling advanced object detection besides simple HSV thresholding.
- **April tag detection** reaches 47Hz, making it feasible for accurate marker-based localization and ID recognition.
- **Visual Odometry (PyrLK)** hits 42Hz, allowing for robust dead-reckoning and motion estimation without wheel encoders.
- **Midas depth estimation** becomes usable at 3.7FPS, unlocking potential for obstacle detection and scene understanding.

These improvements are not just technical metrics—they unlock entirely new strategies:

- Robots can now *localize themselves using fiducial markers* placed on the game table.
- *Object-aware behaviors* become realistic, such as reacting to the presence, position, or type of an object.
- Vision-based *path planning and motion correction* via visual odometry becomes feasible in real-time.

A. Accessibility

We acknowledge that such advancements could risk increasing the gap between experienced and beginner teams. However, our goal is the opposite: by making real-time vision accessible with standard models and off-the-shelf hardware, we aim to *lower the barrier of entry*.

Prior work supports this design philosophy. Educational studies consistently show that simplified development environments - such as Arduino's high-level C++ API or MicroPython on microcontrollers - lead to measurable increases in student engagement, retention, and confidence. For instance, retention rates rose from 71% to 92% in a first-year technical course after a robotics-focused Arduino intervention [18], and classroom studies have shown both faster setup times and significantly higher engagement when using MicroPython on platforms like the Raspberry Pi Pico [19]. These findings align with our intention: empowering teams with minimal setup and immediate results.

By adopting the Raspberry Pi 5 and providing properly scaled sensor units, simplified firmware controls, and full compatibility with the Python ecosystem, our upgrade mirrors these educational gains in a robotics competition context. Even basic strategies - like turning toward a detected marker - become achievable without any machine learning expertise.

This upgrade brings not only performance, but a broader range of creative options to all teams, regardless of experience.

VIII. Conclusion

This work demonstrates a pragmatic upgrade path for the Wombat platform. With minimal cost and disruption, teams gain significant computational headroom and improved control. The firmware and installation guide will be open-sourced to facilitate replication.

References

[1] KISS Institute. *2024 Botball Wombat Build Guide*. 2024.
[2] OpenElab. "Raspberry Pi 3 vs 4 vs 5: Comprehensive Comparison Guide", 2024.
[3] Amazon.com. "Raspberry Pi 5 (8 GB) Product Page", Accessed May 2025.
[4] Element14 Community. "Raspberry Pi 5 GPIO: Is it the Same?", 2024.
[5] Acroname. "All About Back-EMF Motion Control", 2023.
[6] Raspberry Pi Forum. "Raspberry Pi 5 Benchmarks", 2024.
[7] Amazon.com. "Raspberry Pi 3B+ Product Page", Accessed May 2025.
[8] Twozoh. "Micro HDMI to HDMI Adapter", Amazon.com, 2025.
[9] Yahoo! Tech. "Raspberry Pi Price Trends in 2025", 2025.
[10] "Wombat Firmware" GitHub [Online]. Available: https://github.com/kipr/Wombat-Firmware/tree/master Accessed on: June 2, 2025
[11] "Wombat schematic" GitHub [Online]. Available: https://github.com/kipr/Wombat-Firmware/blob/master/docs/Schematics/Wombat%20Lab%20Schematic.pdf Accessed on: June 2, 2025
[12] OpenMV. *OpenMV Cam H7 Specifications*. Available at: https://openmv.io/products/openmv-cam-h7. Accessed June 9, 2025.

[13] OpenMV. *OpenMV Cam H7 Power Consumption*. Available at: https://openmv.io/products/openmv-cam-h7. Accessed June 9, 2025.
[14] Kwabena Agyeman. *Interview on OpenMV H7 Plus Performance*. Adafruit Blog, 2025. Accessed June 9, 2025.
[15] NVIDIA. *NVIDIA Jetson Nano Developer Kit Product Page*. Available at: https://www.amazon.com/NVIDIA-Jetson-Nano-Developer-Kit/dp/B07PZHBDKT. Accessed June 9, 2025.
[16] NVIDIA. *Jetson Nano Revs Up Youth in Worldwide Education*. NVIDIA Blog, 2024. Accessed June 9, 2025.
[17] NVIDIA. *Optimizing YOLOv5 on Jetson Nano*. NVIDIA Developer Forums, 2023. Accessed June 9, 2025.
[18] J. M. de Carvalho *et al.*, "Arduino Baby Project: Robotics as a Tool to Support Permanence and Success of Technical Course Students in Informatics," in *Proc. IEEE Frontiers in Education Conf. (FIE)*, 2021, doi: 10.1109/FIE49875.2021.9637315.
[19] E. Dönmez and S. Gür, "Teaching Embedded Systems and IoT at the University Using MicroPython on Raspberry Pi Pico," in *Proc. ACM Int. Conf. Computing Education*, 2024, doi: 10.1145/3635059.3635079.