Mathematical Formulation of the Training Procedures

Gokalp Cevik

1 Problem Setup

Note Main reference for this section is [1].

We aim to predict the partial pressure of oxygen (pO_2) from observed photoluminescence lifetimes (τ_{obs}), temperatures (T), and pulse counts (n). The core physical relationship is given by the Stern–Volmer equation

$$\tau = \frac{\tau_0}{1 + K_{sv}\, pO_2}, \tag{1}$$

where τ_0 and K_{sv} are material parameters that depend on experimental conditions (e.g. temperature and photo-bleaching). There are modified and improved models to the original Stern–Volmer equation such as Modified S–V, Multi-site Quenching model, etc.

1.1 Underlying Photophysical Principles

Photoluminescence decay is governed by competing pathways that determine the observed lifetime:

- Radiative decay rate (k_r): emission of a photon
- Non-radiative decay rate (k_{nr}): internal processes without photon emission
- Quenching processes: additional non-radiative pathways introduced by quenchers

The lifetime without quencher is determined by intrinsic decay rates:

$$\tau_0 = \frac{1}{k_r + k_{nr}} \tag{2}$$

In the presence of oxygen, an additional decay pathway appears:

$$\tau = \frac{1}{k_r + k_{nr} + k_q[O_2]} \tag{3}$$

Taking the ratio yields the Stern-Volmer relationship:

$$\frac{\tau_0}{\tau} = \frac{k_r + k_{nr} + k_q[O_2]}{k_r + k_{nr}} = 1 + \frac{k_q[O_2]}{k_r + k_{nr}} = 1 + k_q\tau_0[O_2] \tag{4}$$

This establishes $K_{SV} = k_q\tau_0$ as the Stern-Volmer constant, connecting microscopic rate constants to the macroscopic quenching response. The bimolecular quenching constant k_q relates to diffusion through the Smoluchowski equation:

$$k_q = 4\pi N_A(R_F + R_Q)(D_F + D_Q) \tag{5}$$

where:

- N_A: Avogadro's number
- R_F: Radius of the fluorophore molecule
- R_Q: Radius of the quencher molecule (oxygen)

- D_F: Diffusion coefficient of the fluorophore
- D_Q: Diffusion coefficient of the quencher

The diffusion coefficients follow Stokes-Einstein temperature dependence:

$$D = \frac{kT}{6\pi\eta r} \tag{6}$$

where:

- k: Boltzmann constant
- T: Absolute temperature
- η: Viscosity of the medium
- r: Radius of the diffusing molecule

Temperature affects both τ_0 and k_q differently: τ_0 decreases with temperature as non-radiative processes become more efficient, while k_q increases with temperature due to enhanced diffusion.

1.2 Temperature Effects on Decay Processes

Temperature influences both radiative and non-radiative decay processes, creating complex dependencies in oxygen sensing systems.

1.2.1 Radiative Decay Temperature Dependence

The radiative decay rate (k_r) exhibits modest temperature sensitivity through Strickler-Berg relationship:

$$k_r \propto n^2(T) \cdot \int [\varepsilon(\nu)/\nu] d\nu \cdot \int [F(\nu) \cdot \nu^3] d\nu / \int [F(\nu)] d\nu \tag{7}$$

1.2.2 Non-radiative Decay Temperature Dependence

Non-radiative decay processes (k_{nr}) show significantly stronger temperature sensitivity:

- Internal conversion follows activated behavior:

$$k_{IC}(T) = A_{IC} \cdot \exp(-E_{IC}/RT) \tag{8}$$

  with activation energies E_{IC} typically 10-40 kJ/mol

- Intersystem crossing in phosphorescent systems:

$$k_{ISC}(T) = A_{ISC} \cdot \exp(-E_{ISC}/RT) \tag{9}$$

- Vibrational relaxation following approximate relation:

$$k_{vib} \propto T^{1/2} \tag{10}$$

- Thermal activation to quenching states (particularly relevant for metal complexes):

$$k_q(T) = A_q \cdot \exp(-\Delta E/RT) \tag{11}$$

These processes typically cause 5-15% increases in k_{nr} per 10°C, creating much stronger temperature dependence than for radiative processes.

1.3 Relationship Between Quenching Mechanisms and Decay Processes

The Stern-Volmer model encompasses two distinct quenching mechanisms that interact differently with radiative and non-radiative decay pathways.

1.3.1 Dynamic Quenching

Dynamic quenching occurs through collisional interactions between excited fluorophores and oxygen molecules. It introduces an additional non-radiative pathway:

$$k_{total} = k_r + k_{nr} + k_q[\mathrm{O_2}] \tag{12}$$

This mechanism affects both quantum yield and lifetime:

$$\Phi = \frac{k_r}{k_r + k_{nr} + k_q[\mathrm{O_2}]} \tag{13}$$

$$\tau = \frac{1}{k_r + k_{nr} + k_q[\mathrm{O_2}]} \tag{14}$$

The temperature dependence of dynamic quenching efficiency is determined by:

$$\frac{k_q[\mathrm{O_2}]}{k_r + k_{nr}} \propto \frac{T}{\eta(T)} \cdot \exp\left(-E_{diff}/RT\right) \tag{15}$$

where E_{diff} is the activation energy for diffusion.

1.3.2 Static Quenching

Static quenching occurs through formation of non-emissive ground-state complexes between fluorophore and oxygen. Unlike dynamic quenching, it:

- Does not alter decay rates of emitting molecules
- Reduces emitting population without changing their photophysical properties
- Affects quantum yield but not lifetime of remaining emitters

The mathematical relationship for static quenching:

$$\Phi = \Phi_0(1 - f_{complex}) = \Phi_0 \cdot \frac{1}{1 + K_S[\mathrm{O_2}]} \tag{16}$$

$$\tau = \tau_0 \tag{17}$$

where K_S is the association constant for complex formation.
The temperature dependence of static quenching follows:

$$K_S(T) = K_S(T_0) \cdot \exp\left[\frac{-\Delta H}{R}\left(\frac{1}{T} - \frac{1}{T_0}\right)\right] \tag{18}$$

where ΔH is typically negative, causing K_S to decrease with increasing temperature.

1.4 Multi-site Quenching Model Extensions

For heterogeneous systems like polymer-based oxygen sensors, the multi-site quenching model provides a more accurate framework:

$$\frac{\tau_0}{\tau} = \sum_i \frac{f_i}{1 + K_{SV,i}[\mathrm{O_2}]} \tag{19}$$

Each site-specific Stern-Volmer constant can be decomposed into:

$$K_{SV,i} = k_{q,i}\tau_{0,i} \tag{20}$$

The overall decay function becomes:

$$\frac{1}{\tau} = \sum_i f_i(k_{r,i} + k_{nr,i} + k_{q,i}[O_2]) \tag{21}$$

This explains non-linear Stern-Volmer plots observed in practical oxygen sensing systems, as each microenvironment responds differently to temperature, photobleaching, and other environmental factors.

1.5 Practical Implementation Considerations

Commercial oxygen sensing devices typically employ several approaches to address these complex dependencies:

- Multi-parameter calibration matrices accounting for temperature, pressure, and humidity
- Machine learning algorithms to model non-linear relationships
- Two-site models with temperature compensation functions
- Dual-lifetime referencing techniques
- Real-time recalibration protocols

The most robust approaches combine physical models with empirical correction factors to account for the differential effects of temperature on τ_0 and k_q, along with sensor aging and photobleaching effects over time.

1.6 τ_0 and K_{SV} in Practice

In real polymer sensor films, both τ_0 and K_{SV} exhibit complex dependencies on multiple factors:

- Temperature affects τ_0 through Arrhenius-type behavior of non-radiative decay and K_{SV} through changes in diffusion rates
- Photodegradation alters the microenvironment of luminophores over time
- Humidity influences local oxygen solubility and polymer matrix properties
- Manufacturing variability creates heterogeneous sensor sites requiring multi-site models

The multi-site quenching model accounts for heterogeneous distribution of fluorophores in different microenvironments:

$$\frac{\tau_0}{\tau} = \sum_i \frac{f_i}{1 + K_{SV,i}[O_2]} \tag{22}$$

or equivalently:

$$\frac{1}{\tau} = \sum_i f_i \left(\frac{1}{\tau_{0,i}} + \frac{k_{q,i}[O_2]}{\tau_{0,i}} \right) \tag{23}$$

Where:

- f_i: Fractional contribution of site i to the overall emission (where $\sum_i f_i = 1$)
- $K_{SV,i} = k_{q,i}\tau_{0,i}$: Site-specific Stern-Volmer constant
- $\tau_{0,i}$: Unquenched lifetime of fluorophores in site i

- $k_{q,i}$: Bimolecular quenching constant specific to site i, reflecting local accessibility and diffusion properties

Each site may respond differently to temperature, humidity, and aging effects, creating a complex multi-dimensional calibration problem. Commercial systems typically address this by implementing comprehensive calibration matrices or machine learning approaches that can adaptively map lifetime measurements to oxygen concentrations across varying environmental conditions.

2 Network Architectures

2.1 Architecture 1: Stern–Volmer Network

A dedicated MLP g_ϕ predicts the physical parameters

$$g_\phi : (T, n) \longmapsto (\widehat{\tau}_0,\ \widehat{\log K_{sv}},\ \text{offset}). \tag{24}$$

Plugging the outputs into (1) yields an estimated pO_2 which is supervised with an MSE data loss plus optional monotonicity and Arrhenius priors.

2.2 Architecture 2: Joint Network (pO_2 Regressor S–V Prior)

Two MLPs are trained *simultaneously*:

$$f_\theta : (\tau_{\text{obs}}, T, n) \to \widehat{\text{pO}_2}, \qquad \text{(regressor)}$$
$$g_\phi : (T, n) \to (\widehat{\tau}_0, \widehat{\log K_{sv}}, \text{offset}), \qquad \text{(physics)}$$

The regression loss $\mathcal{L}_{\text{data}}$ is augmented with a physics consistency term $\mathcal{L}_{\text{physics}}$ which forces f_θ and g_ϕ to agree with Stern–Volmer through lifetime-ratio matching. Smoothness, monotonicity and Arrhenius penalties regularise the solution.

2.3 Architecture 3: Two-Stage Training (Physics Pre-training → Regressor Alignment)

The Two-Stage architecture is designed to combine the strength of physical priors with data-driven regressors in a stable and scalable manner. It proceeds in two distinct phases:

Stage 1: Train a physics-informed network $g_\phi : (T, n) \to (\widehat{\tau}_0, \widehat{\log K_{sv}}, \text{offset})$ using only temperature and pulse count inputs.

Stage 2: Freeze g_ϕ and train a separate regressor $f_\theta : (\tau_{\text{obs}}, T, n) \to \widehat{\text{pO}_2}$ to predict oxygen partial pressure. A strong physics-consistency loss $\mathcal{L}_{\text{physics}}$ is used to tether f_θ to the predictions of g_ϕ. In the final epochs, g_ϕ can optionally be unfrozen at reduced learning rate for joint fine-tuning.

Regressor Architecture. The regressor f_θ uses a five-layer feedforward network with the following dimensions:

$$\text{Input: 3 features } (\tau_{\text{obs}}, T, \texttt{log1p}(n)) \longrightarrow 128 \to 64 \to 32 \to 16 \to 1.$$

All hidden layers use ReLU activations. The final output is a scalar prediction for pO_2.

Physics MLP Architecture. The physics model g_ϕ is a four-layer MLP with input size 2 (temperature, log1p of pulse count), and output dimensions 3. The network uses ReLU activations between layers:

$$(T, \log 1p(n)) \longrightarrow 64 \to 64 \to 32 \to 3.$$

The output corresponds to $(\widehat{\tau}_0, \widehat{\log K_{sv}}, \text{offset})$. *Note: The offset parameter is a trainable scalar added to the predicted lifetime before ratio computation to stabilize learning.*

Training Dynamics. During Stage 2, f_θ quickly adapts to the manifold defined by the frozen physics model g_ϕ. Since g_ϕ is fixed, it cannot introduce instability, and its gradients are zero. The physics loss $\mathcal{L}_{\text{physics}}$ acts as a strong potential well: if f_θ deviates from the manifold, the error signal becomes large, pushing predictions back towards consistency.

This architecture was ultimately selected for deployment on the embedded microcontroller.

2.4 Architecture 4: Modified Stern–Volmer (dynamic static quenching)

The classical Stern–Volmer expression in (1) assumes *purely dynamic* collisional quenching. Experimentally, many luminophores also exhibit a *static* contribution caused by ground-state complex formation. Lehrer's modified SV model incorporates both mechanisms:

$$\tau \;=\; \frac{\tau_0}{\big(1 + K_{sv}\,\mathrm{pO_2}\big)\big(1 + K_s\,\mathrm{pO_2}\big)}, \tag{25}$$

where K_s is the static quenching constant. We parameterise

$$g_\phi^{(\mathrm{MSV})} : (T, n) \to (\widehat{\tau}_0,\ \widehat{\log K_{sv}},\ \widehat{\log K_s},\ \text{offset}),$$

and obtain $\mathrm{pO_2}$ by numerically inverting (25) (Newton–Raphson, 10 iterations, see `train_msv.py`). Training mirrors Architecture 1 but adds $\mathcal{L}_{\text{mono}}\big(\partial_T K_s\big)$ and an identical Arrhenius prior on τ_0. The model delivers markedly smaller bias above $\sim$80% O_2 without hurting in-range accuracy.

2.5 Architecture 5: Multi-site Quenching

Polymer matrices often host multiple independent quenching sites. The two-site model expresses the lifetime ratio as

$$\frac{\tau}{\tau_0} \;=\; \frac{f_1}{1 + K_{sv1}\,\mathrm{pO_2}} \;+\; \frac{1 - f_1}{1 + K_{sv2}\,\mathrm{pO_2}}, \tag{26}$$

with fractional contribution $f_1 \in [0, 1]$. The corresponding physics MLP (see `train_multisite.py`)

$$g_\phi^{(\mathrm{MSQ})} : (T, n) \to (\widehat{\tau}_0,\ \widehat{\log K_{sv1}},\ \widehat{\log K_{sv2}},\ \widehat{f_1},\ \text{offset})$$

is trained with the same monotonicity and Arrhenius priors as above. Equation (26) is inverted with a damped Newton-like solver; the derivative in the physics-loss term is computed analytically, so back-propagation remains stable.

3 Loss Functions

For a batch of N samples ($i = 1, \ldots, N$) we use

$$\begin{aligned}
\mathcal{L}_{\text{data}} &= \frac{1}{N}\sum_i \big(\widehat{\mathrm{pO_2}^{(i)}} - \mathrm{pO_2}^{(i)}\big)^2, \\
\mathcal{L}_{\text{physics}} &= \frac{1}{N}\sum_i \big(r_{\text{pred}}^{(i)} - r_{\text{meas}}^{(i)}\big)^2, \\
\mathcal{L}_{\text{mono}} &= \mathbb{E}\big[\mathrm{ReLU}(\partial_T \widehat{\tau}_0) + \mathrm{ReLU}(\partial_T \widehat{\log K_{sv}})\big], \\
\mathcal{L}_{\text{arrh}} &= \frac{1}{N}\sum_i \big(\log \widehat{\tau}_0^{(i)} - a/(T^{(i)} + 273.15)\big)^2, \\
\mathcal{L}_{\text{smooth}} &= \mathbb{E}\big[(\partial_{\tau_{\text{obs}}} \widehat{\mathrm{pO_2}})^2\big].
\end{aligned}$$

The total loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda_{\text{phys}}\mathcal{L}_{\text{physics}} + \lambda_{\text{mono}}\mathcal{L}_{\text{mono}} + \lambda_{\text{arrh}}\mathcal{L}_{\text{arrh}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}. \tag{27}$$

In Stage 2, only f_θ receives gradient updates until the optional late unfreezing.

4 Training Procedures and Dynamics

Physics-Only (Stage 1)

We minimise $\mathcal{L}_{\text{data}}$ (plus priors) with Adam (lr = 1×10^{-4}, weight-decay 1×10^{-2}, batch size 256). Early-stopping on validation MSE is used to avoid over-fitting.

Joint Training

All parameters of f_θ and g_ϕ are updated in parallel each step.

Two-Stage Training

During Stage 2 the regressor quickly adapts to the physics manifold provided by g_ϕ. Because g_ϕ is frozen, its gradients are zero and no destructive interference occurs. The physics loss acts as a strongly sloped potential well: if f_θ proposes a non-physical pO_2 the loss and its gradient explode, pulling the prediction back. After convergence we unfreeze g_ϕ for fine-tuning; using a lower learning rate prevents catastrophic forgetting of the physical priors.

5 Implementation Notes

- Gradients of $\mathcal{L}_{\text{smooth}}$ are obtained via automatic differentiation on τ_{obs}.
- Lifetime and temperature are de-normalised *inside* the loss functions to keep the implementation invariant to feature scaling.
- All models are implemented in PyTorch 2.6.0; training scripts are provided under `src/ml/`.

6 Hardware

The hardware platform used is highlighted in Figs. 1, 2.

The microcontroller used in the board is STM32WB35 [2].

7 Dataset, Training, Results & Discussion

7.1 Dataset Acquisition Setup

To acquire the training dataset, a gas testbench setup was used, as highlighted in Figs. 3, 4.

The ground truth values for pO2 was controlled using a Mass Flow Controller (MFC), and the measurements for the lifetime (decay curves), temperature and number of applied pulses for that film was recorded to the computer as raw log files. A Python script was then used to convert the log files into a more friendly/usable JSON formatted files.

7.2 Entire Dataset Distribution

The distribution of the data set that was collected to train the network is seen in Fig. 5. Most of the data collected was relevant to the human psyiological range (i.e. 5.0%-15.0% O2), in addition, I have also collected additional data beyond 20% O2, although in larger steps.

As a limitation of the lab equipment, I had no way of controlling the temperature explicitly, and I relied the lab's AC to give me some variation on the temperature. The data collected had temperatures on the range of 22C and 28C, with little variation in between.

Temperature affects the training and prediction results significantly, thus for better results, there is need to explicitly vary the temperature and augment the data set accordingly for future work.

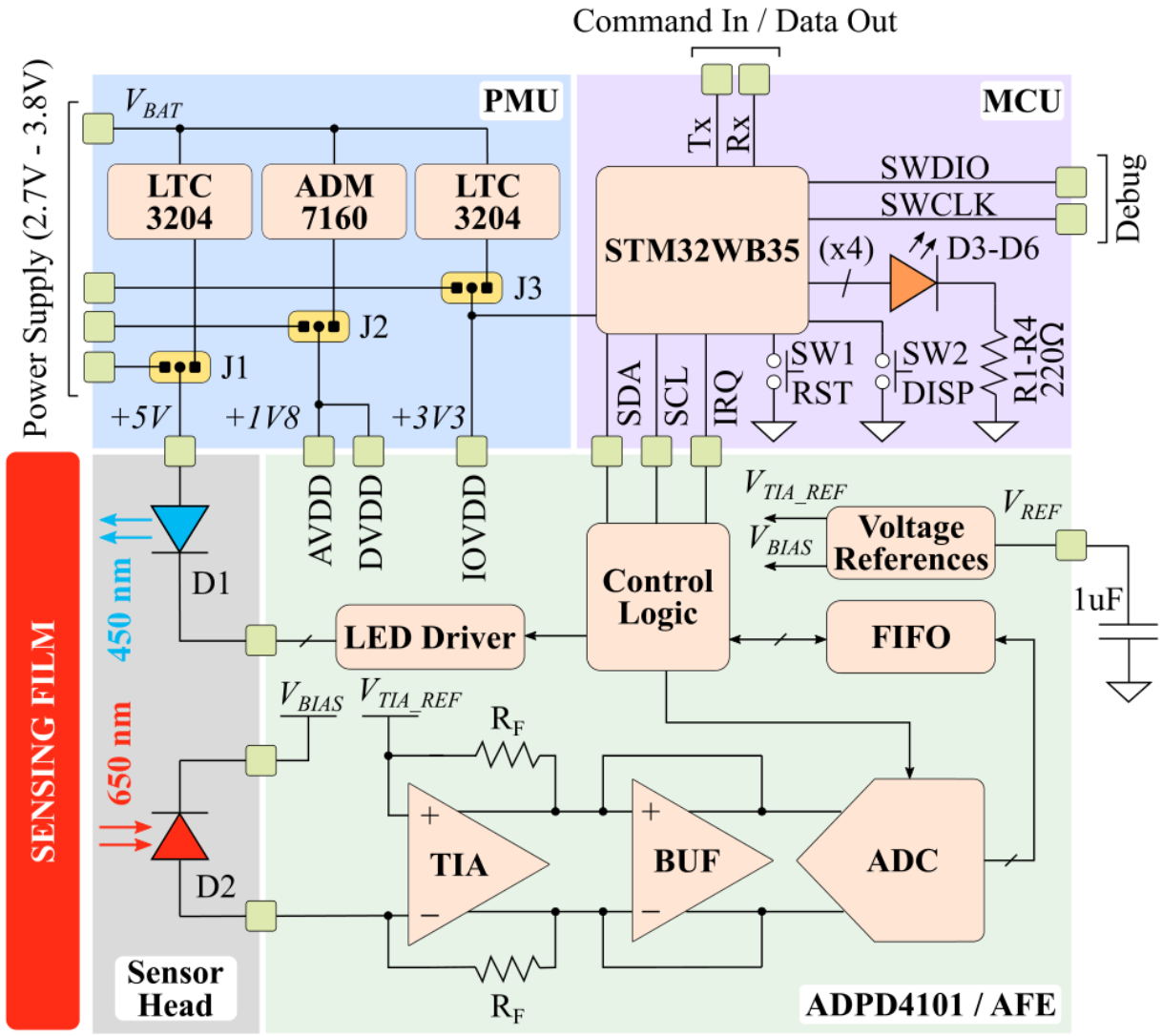

Figure 1: Block Diagram of the Hardware Platform[3]

7.3 Final Model Architecture Accuracy

For field testing (actual unseen data, running on the microcontroller), I decided go with the two-stage architecture, as that yielded the best results in terms of physics and the actual fit performance. I decided to turn-off the Arrhenius and monotonicity penalties completely, as the dynamics of the physics seems to be more complex than what these models suggest. I used $\lambda_{\text{phys}} = 25$ for the physics loss and $\lambda_{\text{smoothness}} = 0.01$ for the smoothness. The fit results for the test dataset is seen in Fig. 6 and the actual performance on the microcontroller (with controlled O2 percentage) is seen in Fig. 7, with fit and accuracy metrics highlighted in the figiure.

The quantized model (using PyTorch's "fbgemm" quantization scheme and using post-training quantization (PTQ)), yielded very similar accuracy compared to the 32-bit floating point model, where the comparison figure and the relevant metrics is depicted in Fig. 8.

8 Deviation from the Stern–Volmer Ideal at High Oxygen Fractions

Training window and physical context. Most of my dataset used for model development lie below 23% O_2 ($pO_2 \sim 17$ cmHg). The classical linear Stern–Volmer (SV) relation $\tau = \tau_0/(1 + K_{sv}\, pO_2)$ continues to describe

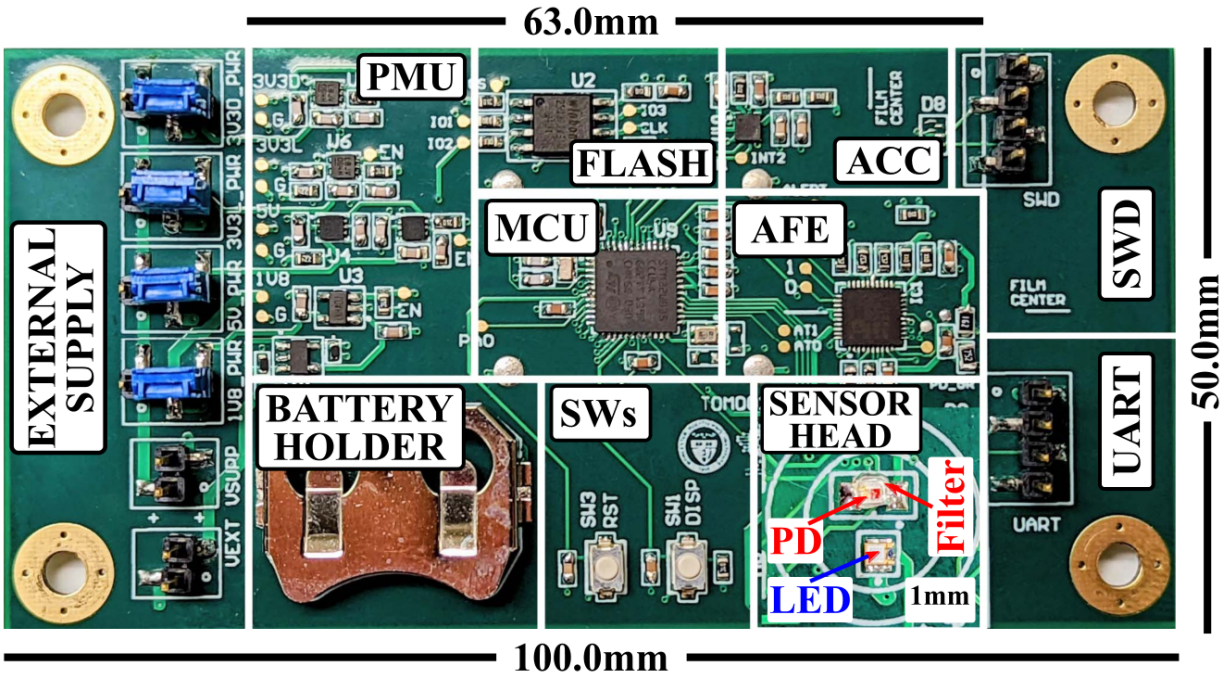

Figure 2: Board View of the Hardware Platform[3]

most luminophore systems accurately up to $\approx$ 80% O_2. Only in the *very* high-oxygen regime (80–100%) does experimental evidence show systematic curvature—either an upward bend (sub-quenching) or a downward bend (supra-quenching), depending on dye and matrix.

Behaviour beyond 23% O_2.

- **Physics-only (Architecture 1).** Extrapolates strictly linearly because pO_2 is inferred *exactly* from the SV formula. Numerical stability is excellent, but the model cannot reproduce the curvature observed above 80% O_2, so absolute error increases monotonically in that regime. There is great dependence on the temperature, so additional correction algorithms or much larger temperature variation within the training dataset is needed.
- **Joint model (Architecture 2).** The data-driven regressor f_θ is tethered to physics only by a λ_{phys}-weighted loss term. Once lifetimes fall outside the labelled manifold, f_θ is free to learn non-SV mappings. The result is a bad extrapolation since the regressor has not seen any data beyond 23%, except at a few hundred points at every 10% after 30%.
- **Two-stage model (Architecture 3).** Pretraining and freezing the physics network g_ϕ provides a stronger prior. The behaviour is much better than the joint model, since the trained physical model injects a strong numerically stable prior, and the network is able to learn from the data in a much more stable way.

Note. Using the purely physical model, we can augment the dataset up to 70-80% (i.e. generate synthetic data using the physical model, Stern-Volmer equation), however this is not ideal. The most reliable approach would be collecting more data beyond 23%, ideally at least in 1% steps and with greatly varying temperature (relevant human range: possibly 25-36C). We currently do not have a reliable way of varying the temperature.

Practical implication. For applications expected to encounter oxygen fractions above 80%, either (i) acquire a small calibration set in that regime or (ii) incorporate higher-order quenching terms (e.g. two-site or modi-

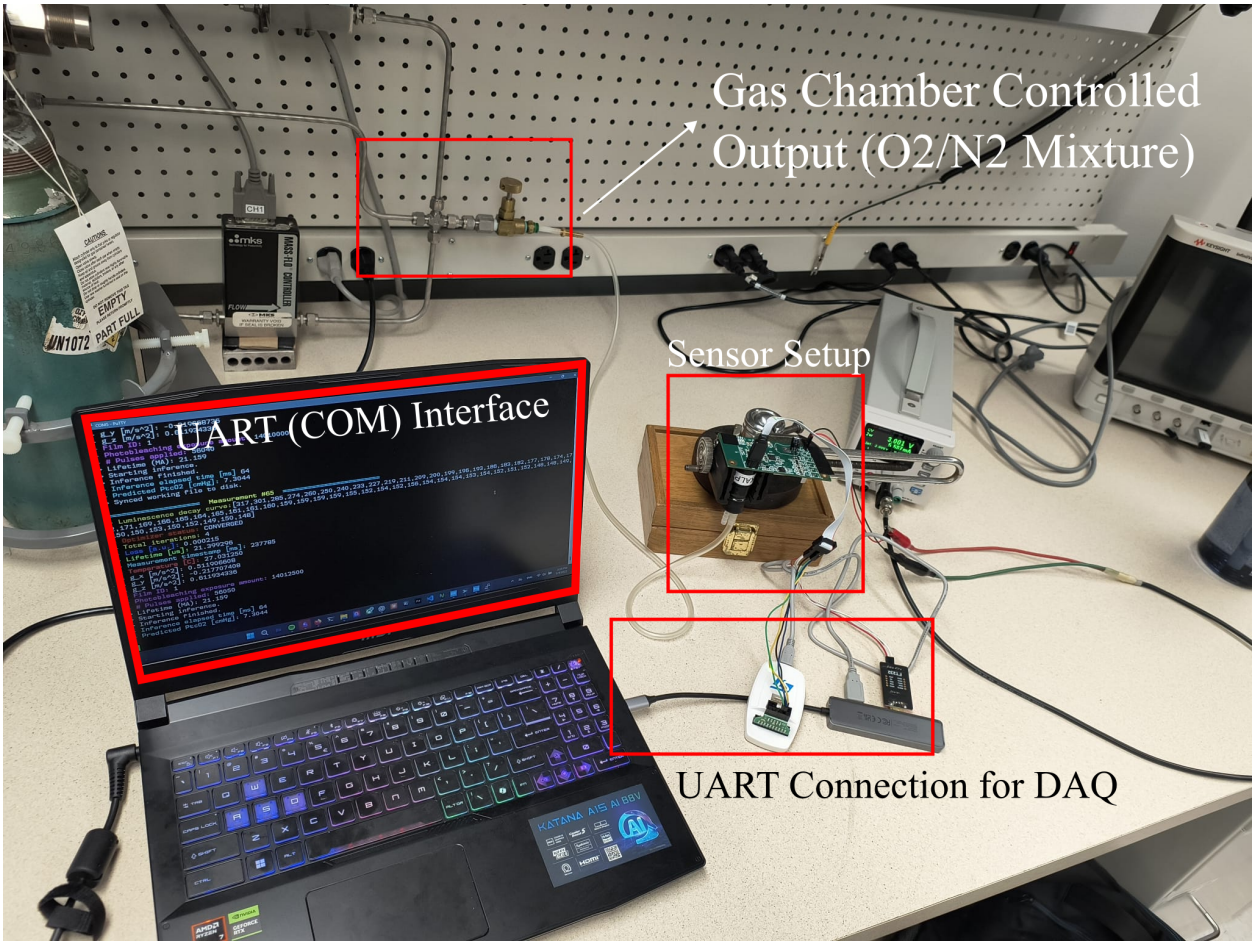

Figure 3: Data Acquisition Setup for Dataset and Testing

fied SV models) so that the inference network can follow the empirical curvature without sacrificing in-range accuracy.

9 Microcontroller Deployment

For deployment on constrained embedded hardware, two versions of the trained model were implemented: a 32-bit floating point (FP32) variant and an 8-bit quantized (int8) variant using PyTorch's post-training quantization (PTQ) with the *fbgemm* backend. Both implementations were tested on a Cortex-M-based microcontroller, operating without an operating system and using the FatFS library to load model parameters from external flash storage.

Floating Point Inference

The FP32 version executes a five-layer MLP of shape $[3 \to 128 \to 64 \to 32 \to 16 \to 1]$. Each layer is streamed from flash by reading one weight row and its corresponding bias value at a time. This enables inference without allocating memory for the full model, significantly reducing RAM requirements. Activations are passed layer-by-layer using a shared input/output buffer, with ReLU applied after all hidden layers.

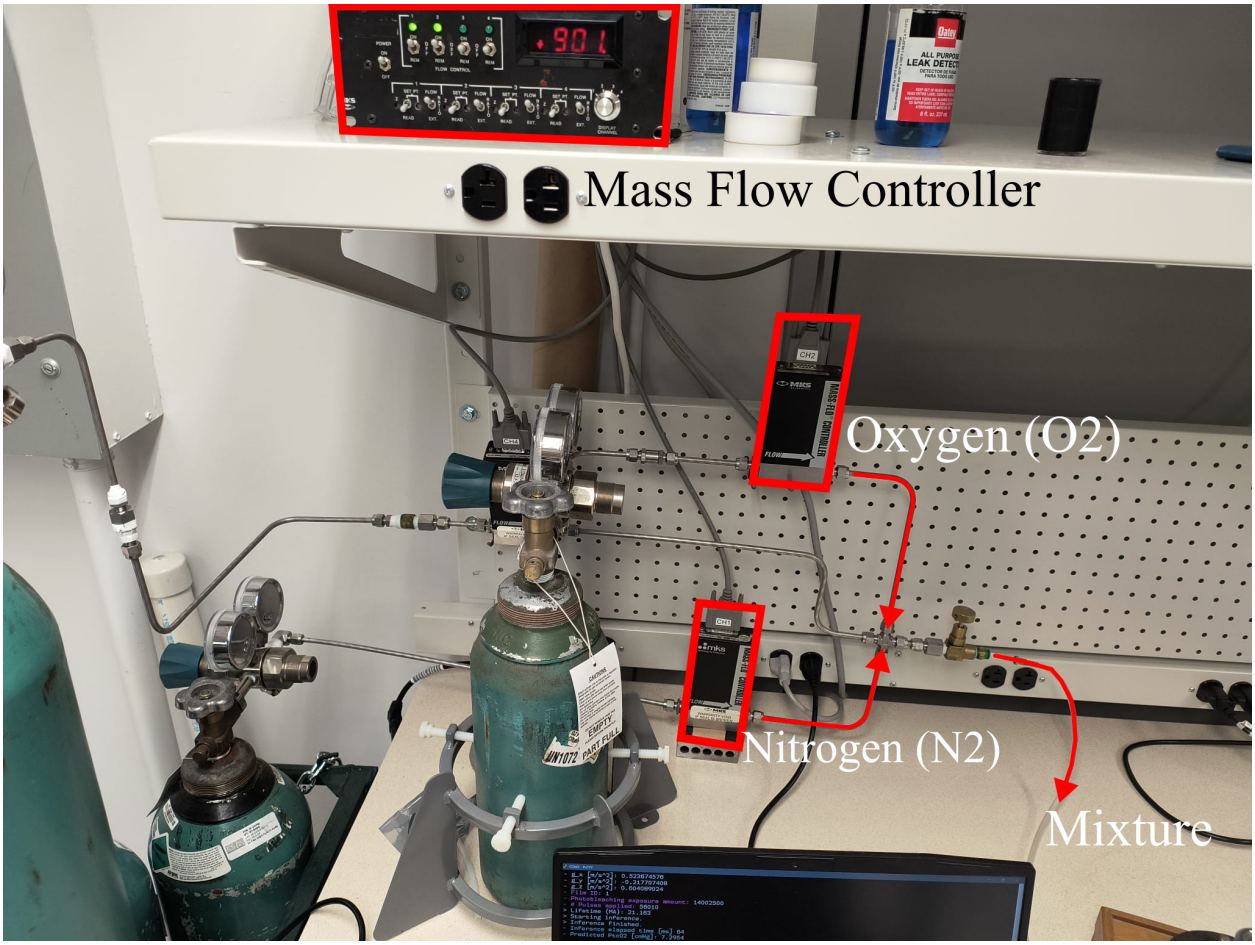

Figure 4: Mass Flow Controller

Quantized Inference

For the quantized model, weights are stored as int8 with symmetric per-channel quantization, and activations are stored as uint8 in RAM with `reduce_range=True`, so `int8_t` range is sufficient. All quantization parameters (scales, zero-points) and weights are stored in separate binary files on the external flash memory and streamed from flash during inference the same way as the floating point model.

Pruning

I have not had time to implement or look into pruning.

Model Accuracy

The quantized model achieves near-identical performance to its floating-point counterpart, as shown in Fig. 8.

Latency

The latency is dominated by the flash read speed; inference time is approximately 64 ms for both versions. This makes sense because on Cortex-M4F units, floating point and integer mul/add operations take a single cycle, once the pipeline is full.

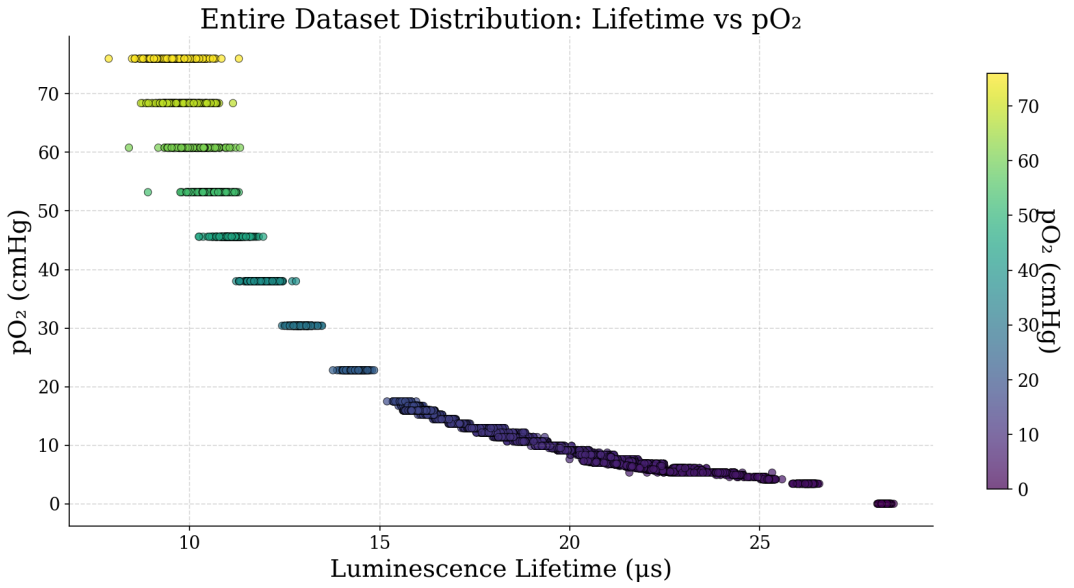

Figure 5: Entire Dataset Distribution ($PtcO_2$ vs Lifetime

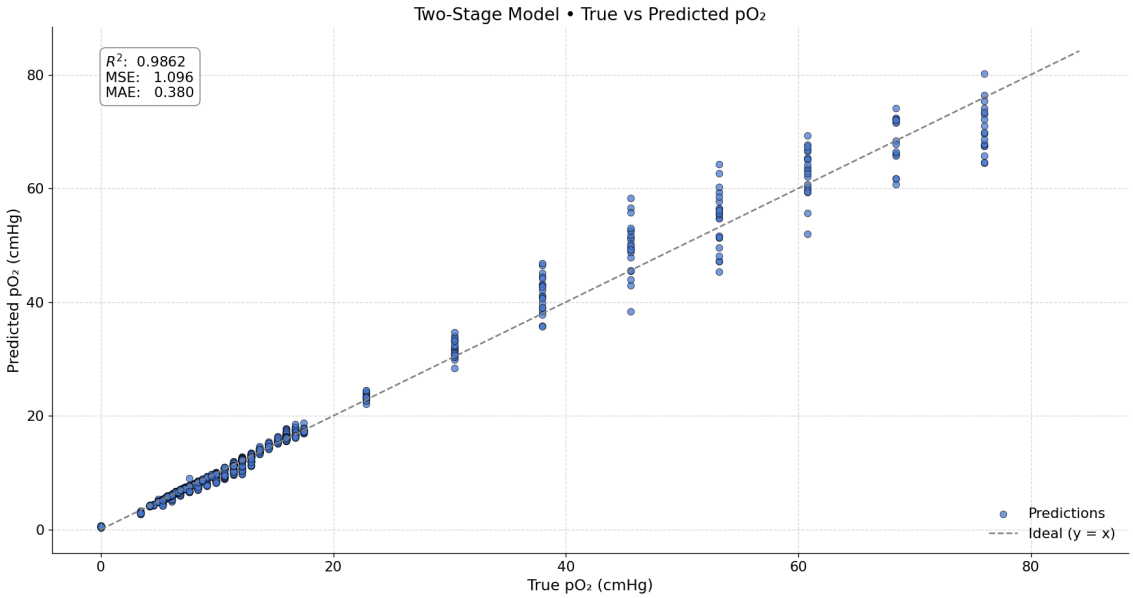

Figure 6: Test Dataset Performance of the Two-Stage Architecture

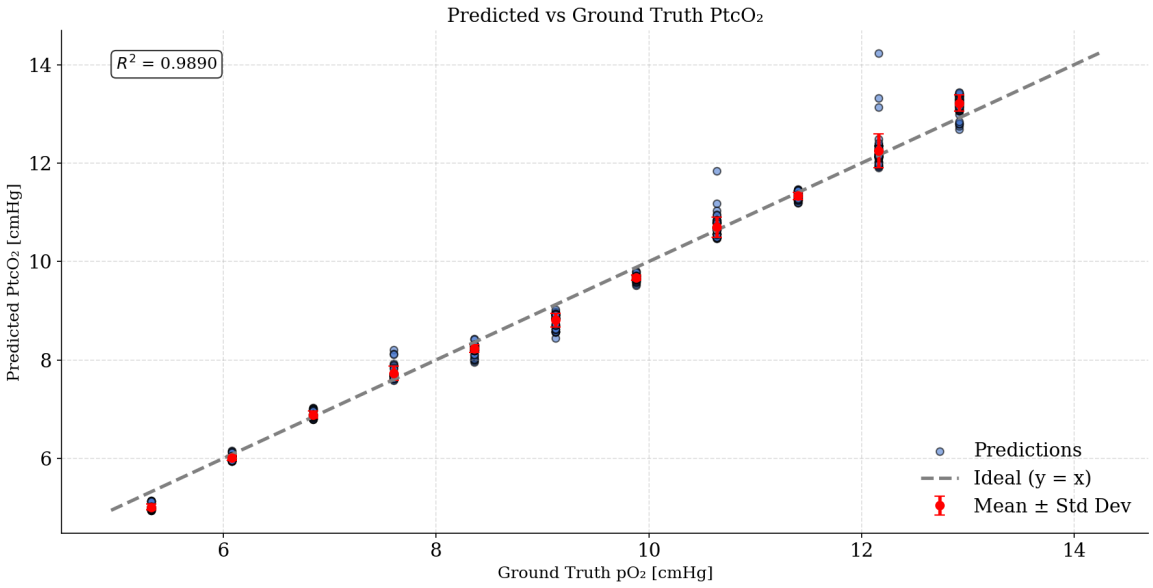

Figure 7: Field Performance (running on the microcontroller) of the Two-Stage Architecture

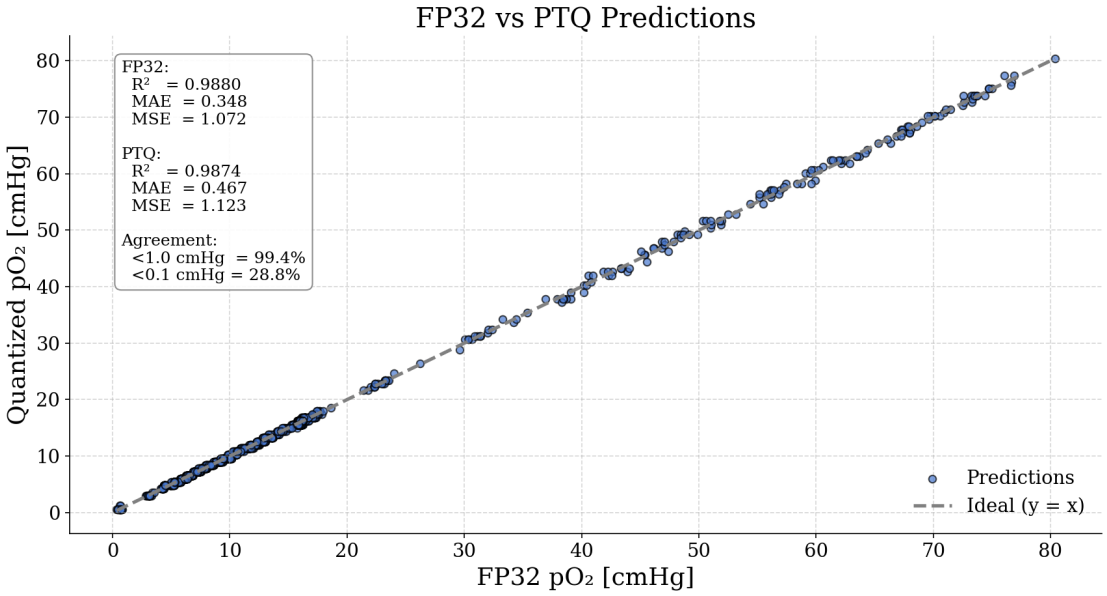

Figure 8: Comparison of the Quantized (post-training quantization) and the 32-bit Floating Point Model

Energy

The microcontroller power consumption jumps to around 30 mW during inference. Considering each inference takes around 64 ms, the total energy per inference is around 1.92 mJ.

Implementation Notes

- Each weight row is read on-the-fly to minimize memory usage.
- Layer activation buffers are reused between layers with scratch buffers.
- All inference code is written in C++, with FatFS used to handle file I/O from a mounted SPI NOR flash.

References

[1] Joseph R. Lakowicz. *Principles of fluorescence spectroscopy*. Jan. 2006. DOI: 10.1007/978-0-387-46312-4. URL: https://doi.org/10.1007/978-0-387-46312-4.

[2] STMicroelectronics. *STM32WB35CC Datasheet and Reference Manual*. https://www.st.com/en/microcontrollers-microprocessors/stm32wb35cc.html (accessed: Oct. 21, 2024). URL: https://www.st.com/en/microcontrollers-microprocessors/stm32wb35cc.html.

[3] Vladimir Vakhter et al. "A Prototype Wearable Device for Noninvasive Monitoring of Transcutaneous Oxygen". In: *IEEE Transactions on Biomedical Circuits and Systems* 17.2 (2023), pp. 323–335. DOI: 10.1109/TBCAS.2023.3251321.