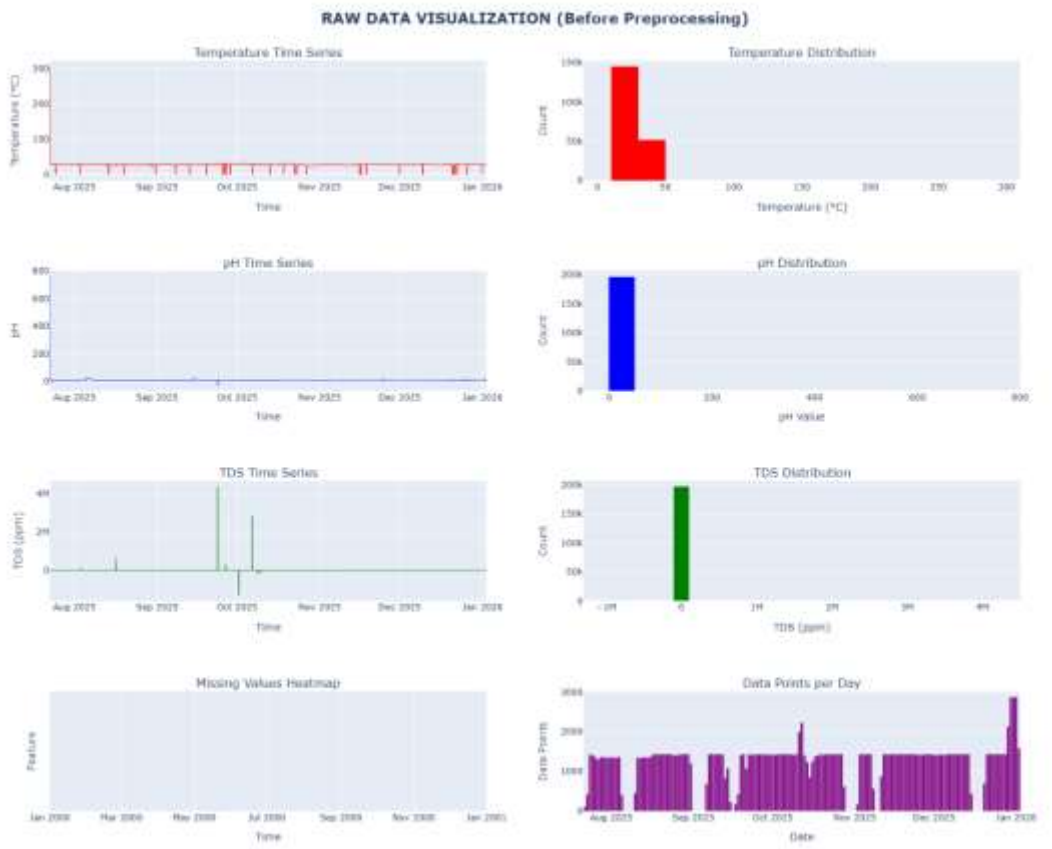
APPENDICES

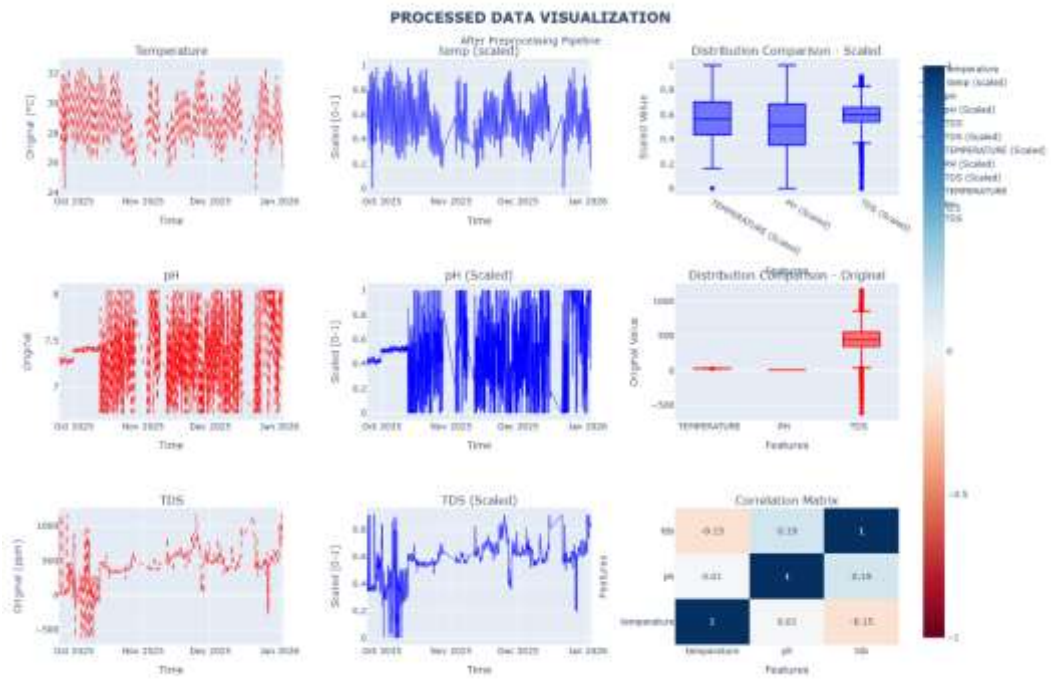
Appendix 1. Preview of Raw Data in Graph



Appendix 2. Preview of Raw Data in Table

id	waktu	water_level_status	water_temp_c	tds_ppm	ph_value
1	22/07/2025 18:52	Air Krg 1cm	30.625	318.385	754.187
2	22/07/2025 18:53	Air Krg 1cm	30.625	318.574	757.656
3	22/07/2025 18:59	Tinggi Normal	30.625	317.82	753.531
4	22/07/2025 19:02	Tinggi Normal	30.625	318.762	755.219
5	22/07/2025 19:07	Air Krg 1cm	30.625	318.951	755.969
6	22/07/2025 19:09	Air Krg 1cm	30.625	318.574	755.969
7	22/07/2025 19:11	Air Krg 1cm	305.625	318.103	754.187

Appendix 3. Preview of Data Processed in Graph



Appendix 4. Preview of Unscaled Version Dataset Prepared

time	temperature	ph	tds	date	temperature_original	ph_original	tds_original
2025-09-24 11:00:00	0.5250755287	0.4204545455	0.0070989115	2025-09-24	28.63	7.28	8.5
2025-09-24 11:30:00	0.5770392749	0.4507575758	0.003786086133	2025-09-24	29.06	7.32	5
2025-09-24 12:00:00	0.6229607251	0.4507575758	0.0028395646	2025-09-24	29.44	7.32	4
2025-09-24 12:30:00	0.6531722054	0.4507575758	0.003786086133	2025-09-24	29.69	7.32	5
2025-09-24 13:00:00	0.6833836858	0.4507575758	0.01514434453	2025-09-24	29.94	7.32	17
2025-09-24 13:30:00	0.7208459215	0.4507575758	0.0198769522	2025-09-24	30.25	7.32	22
2025-09-24 14:00:00	0.7208459215	0.4356060606	0.0227165168	2025-09-24	30.25	7.3	25
2025-09-24 14:30:00	0.7438066465	0.4431818182	0.003786086133	2025-09-24	30.44	7.31	5
2025-09-24 15:00:00	0.7583081571	0.4507575758	0.8287742546	2025-09-24	30.56	7.32	876.6
2025-09-24 15:30:00	0.7740181269	0.4507575758	0.9455750118	2025-09-24	30.69	7.32	1057.5

Appendix 5. Preview of in Scaled Version Dataset Prepared

time	temperature_scaled	ph_scaled	tds_scaled
2025-09-24 11:00:00	0.5250755287	0.4204545455	0.0070989115
2025-09-24 11:30:00	0.5770392749	0.4507575758	0.003786086133
2025-09-24 12:00:00	0.6229607251	0.4507575758	0.0028395646
2025-09-24 12:30:00	0.6531722054	0.4507575758	0.003786086133
2025-09-24 13:00:00	0.6833836858	0.4507575758	0.01514434453
2025-09-24 13:30:00	0.7208459215	0.4507575758	0.0198769522
2025-09-24 14:00:00	0.7208459215	0.4356060606	0.0227165168
2025-09-24 14:30:00	0.7438066465	0.4431818182	0.003786086133

Appendix 6. Preview of Input Sequence

sequence_id	start_time	target_time	t-143	t-142	t-141	t-140	t-139	t-138
1	2025-09-24 11:00:00	2025-09-27 11:00:00	[28.63, 7.26, 8]	[29.06, 7.32, 5]	[29.44, 7.32, 4]	[29.69, 7.32, 5]	[29.94, 7.32, 17]	[30.25, 7.32, 22]
2	2025-09-24 11:30:00	2025-09-27 11:30:00	[29.06, 7.32, 5]	[29.44, 7.32, 4]	[29.69, 7.32, 5]	[29.94, 7.32, 17]	[30.25, 7.32, 22]	[30.25, 7.30, 25]
3	2025-09-24 12:00:00	2025-09-27 12:00:00	[29.44, 7.32, 4]	[29.69, 7.32, 5]	[29.94, 7.32, 17]	[30.25, 7.32, 22]	[30.25, 7.30, 25]	[30.44, 7.31, 5]
4	2025-09-24 12:30:00	2025-09-27 12:30:00	[29.69, 7.32, 5]	[29.94, 7.32, 17]	[30.25, 7.32, 22]	[30.25, 7.30, 25]	[30.44, 7.31, 5]	[30.56, 7.32, 877]
5	2025-09-24 13:00:00	2025-09-27 13:00:00	[29.94, 7.32, 17]	[30.25, 7.32, 22]	[30.25, 7.30, 25]	[30.44, 7.31, 5]	[30.56, 7.32, 877]	[30.69, 7.32, 1058]
6	2025-09-24 13:30:00	2025-09-27 13:30:00	[30.25, 7.32, 22]	[30.25, 7.30, 25]	[30.44, 7.31, 5]	[30.56, 7.32, 877]	[30.69, 7.32, 1058]	[30.75, 7.28, 1058]
7	2025-09-24 14:00:00	2025-09-27 14:00:00	[30.25, 7.30, 25]	[30.44, 7.31, 5]	[30.56, 7.32, 877]	[30.69, 7.32, 1058]	[30.75, 7.28, 1058]	[30.81, 7.29, 1058]
8	2025-09-24 14:30:00	2025-09-27 14:30:00	[30.44, 7.31, 5]	[30.56, 7.32, 877]	[30.69, 7.32, 1058]	[30.75, 7.28, 1058]	[30.81, 7.29, 1058]	[30.88, 7.30, 1058]
9	2025-09-24 15:00:00	2025-09-27 15:00:00	[30.56, 7.32, 877]	[30.69, 7.32, 1058]	[30.75, 7.28, 1058]	[30.81, 7.29, 1058]	[30.88, 7.30, 1058]	[30.81, 7.28, 528]
10	2025-09-24 15:30:00	2025-09-27 15:30:00	[30.69, 7.32, 1058]	[30.75, 7.28, 1058]	[30.81, 7.29, 1058]	[30.88, 7.30, 1058]	[30.81, 7.28, 528]	[30.75, 7.30, 169]
11	2025-09-24 16:00:00	2025-09-27 16:00:00	[30.75, 7.28, 1058]	[30.81, 7.29, 1058]	[30.88, 7.30, 1058]	[30.81, 7.28, 528]	[30.75, 7.30, 169]	[30.63, 7.28, 87]
12	2025-09-24 16:30:00	2025-09-27 16:30:00	[30.81, 7.29, 1058]	[30.88, 7.30, 1058]	[30.81, 7.28, 528]	[30.75, 7.30, 169]	[30.63, 7.28, 87]	[30.50, 7.27, 17]

Appendix 7. Preview of Output Sequence

sequence_id	temp_t+1	ph_t+1	tds_t+1	temp_t+2	ph_t+2	tds_t+2	temp_t+3	ph_t+3
1	28.94	7.33	52	29.31	7.34	6	29.88	7.3
2	29.31	7.34	6	29.88	7.34	32	30.19	7.34
3	29.88	7.34	32	30.19	7.34	14	30.56	7.32
4	30.19	7.34	14	30.56	7.32	441	30.88	7.32
5	30.56	7.32	441	30.88	7.32	867	31.13	7.29
6	30.88	7.32	867	31.13	7.29	1058	31.44	7.28
7	31.13	7.29	1058	31.44	7.28	1058	31.63	7.27
8	31.44	7.28	1058	31.63	7.27	1058	31.75	7.3
9	31.63	7.27	1058	31.75	7.3	1058	31.75	7.3
10	31.75	7.3	1058	31.75	7.3	1058	31.69	7.29

Appendix 8. Link of All Datasets and Data Visualization

https://go.undiksha.ac.id/dataset_LSTM_livinglab

Appendix 9. Source Code

https://github.com/cindyhps/lstm_forecast_livinglab

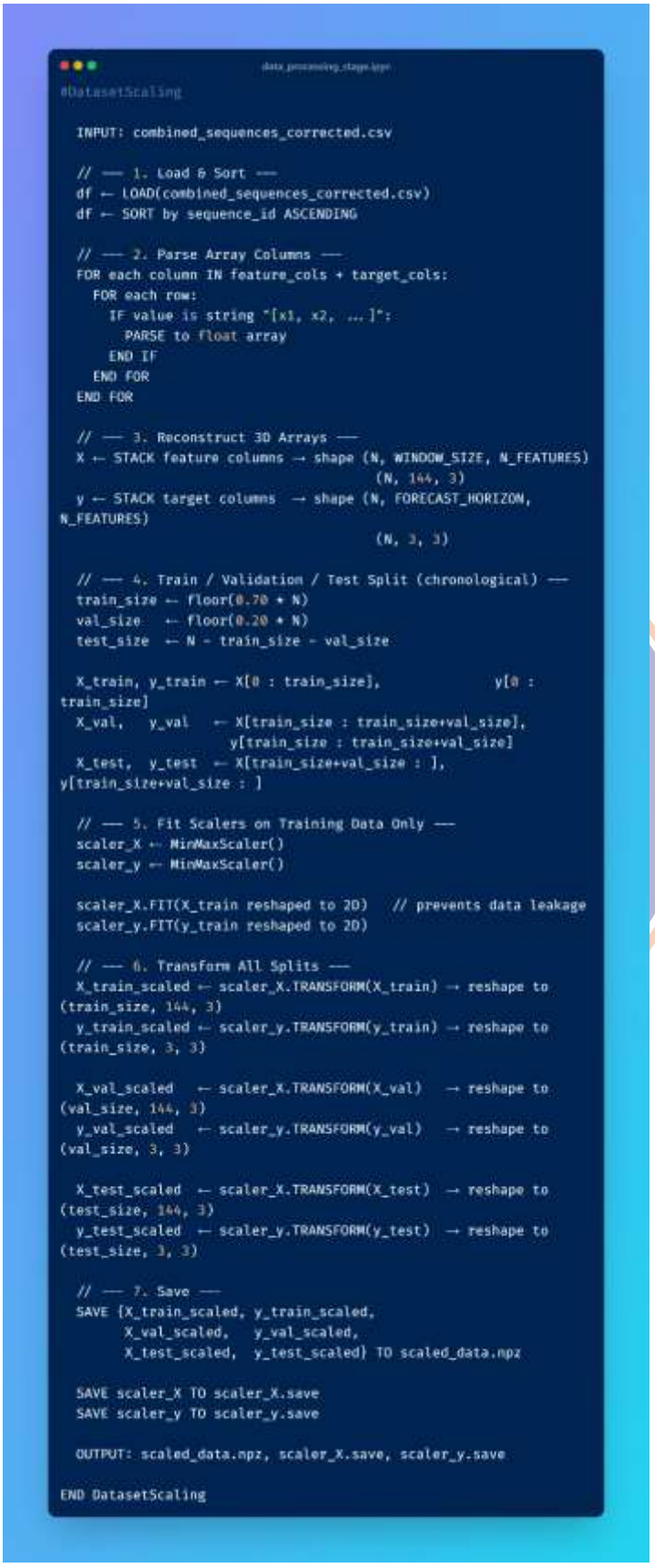
Appendix 10. Code Snippet of LSTM Model Architecture

```
data_processing_stage.ipynb
AAlpha Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam

# Example configuration (can be adjusted per variable)
units = 16          # e.g., temperature
dropout_rate = 0.1
learning_rate = 0.001
output_dim = 1      # single target prediction

model = Sequential([
    LSTM(units, input_shape=(window_size, n_features)),
    Dropout(dropout_rate),
    Dense(output_dim)
])

model.compile(
    optimizer=Adam(learning_rate=learning_rate),
    loss="mse",
    metrics=["mae"]
)

model.summary()
```

Appendix 11. Code Snippet of Data Preprocessing Stage

```
data_processing_stage.ipynb
#DataPreprocessing

  INPUT: raw_sensor_data (CSV file)

  // ── 1. Load & Clean Raw Data ──
  df ← LOAD(raw_sensor_data)
  df ← RENAME columns {waktu → time, water_temp_c →
temperature,
                       ph_value → ph, tds_ppm → tds}
  df ← KEEP columns [time, temperature, ph, tds]
  df ← CONVERT [temperature, ph, tds] TO float

  // ── 2. Filter by Time Interval ──
  df ← FILTER rows WHERE minute(time) IN [0, 30]
  df ← SORT by time ASCENDING
  df ← DROP DUPLICATE timestamps

  // ── 3. Handle Missing Values ──
  FOR each column IN [temperature, ph, tds]:
    FILL missing values using forward-fill, then backward-fill
  END FOR

  // ── 4. Outlier Detection & Removal ──
  FOR each column IN [temperature, ph, tds]:
    q1, q3 ← COMPUTE 25th and 75th percentile
    iqr    ← q3 - q1
    lower  ← q1 - 1.5 * iqr
    upper  ← q3 + 1.5 * iqr
    REMOVE rows WHERE value < lower OR value > upper
  END FOR

  // ── 5. Select Last N Days ──
  cutoff ← max(time) - TOTAL_DAYS_HISTORY (90 days)
  df     ← FILTER rows WHERE time ≥ cutoff
  df     ← SELECT last DATA_POINTS_TOTAL (4320) rows

  // ── 6. Sliding Window Sequence Generation ──
  sequences ← []

  FOR i FROM 0 TO (len(df) - WINDOW_SIZE - FORECAST_HORIZON) STEP
STEP_SIZE:
    window   ← df[i : i + WINDOW_SIZE]          // lookback =
144 steps
    targets  ← df[i + WINDOW_SIZE :
                  i + WINDOW_SIZE + FORECAST_HORIZON]  // t+1,
t+2, t+3

    FOR each feature f IN [temperature, ph, tds]:
      feature_sequence[f] ← window[f]   // shape: (144,)
      target_sequence[f]  ← targets[f]  // shape: (3,)
    END FOR

    sequences.APPEND({feature_sequences, target_sequences,
sequence_id: i})
  END FOR

  // ── 7. Save ──
  SAVE sequences TO combined_sequences_corrected.csv

  OUTPUT: combined_sequences_corrected.csv

END DataPreprocessing
```

Appendix 12. Code Snippet of Data Scalling Stage



```
#DatasetScaling

  INPUT: combined_sequences_corrected.csv

  // ── 1. Load & Sort ──
  df ← LOAD(combined_sequences_corrected.csv)
  df ← SORT by sequence_id ASCENDING

  // ── 2. Parse Array Columns ──
  FOR each column IN feature_cols + target_cols:
    FOR each row:
      IF value is string "[x1, x2, ...]":
        PARSE to float array
      END IF
    END FOR
  END FOR

  // ── 3. Reconstruct 3D Arrays ──
  X ← STACK feature columns → shape (N, WINDOW_SIZE, N_FEATURES)
                                (N, 144, 3)
  y ← STACK target columns  → shape (N, FORECAST_HORIZON,
N_FEATURES)
                                (N, 3, 3)

  // ── 4. Train / Validation / Test Split (chronological) ──
  train_size ← floor(0.70 * N)
  val_size   ← floor(0.20 * N)
  test_size  ← N - train_size - val_size

  X_train, y_train ← X[0 : train_size],          y[0 :
train_size]
  X_val,   y_val   ← X[train_size : train_size+val_size],
                     y[train_size : train_size+val_size]
  X_test,  y_test  ← X[train_size+val_size : ],
y[train_size+val_size : ]

  // ── 5. Fit Scalers on Training Data Only ──
  scaler_X ← MinMaxScaler()
  scaler_y ← MinMaxScaler()

  scaler_X.FIT(X_train reshaped to 2D)   // prevents data leakage
  scaler_y.FIT(y_train reshaped to 2D)

  // ── 6. Transform All Splits ──
  X_train_scaled ← scaler_X.TRANSFORM(X_train) → reshape to
(train_size, 144, 3)
  y_train_scaled ← scaler_y.TRANSFORM(y_train) → reshape to
(train_size, 3, 3)

  X_val_scaled   ← scaler_X.TRANSFORM(X_val)   → reshape to
(val_size, 144, 3)
  y_val_scaled   ← scaler_y.TRANSFORM(y_val)   → reshape to
(val_size, 3, 3)

  X_test_scaled  ← scaler_X.TRANSFORM(X_test)  → reshape to
(test_size, 144, 3)
  y_test_scaled  ← scaler_y.TRANSFORM(y_test)  → reshape to
(test_size, 3, 3)

  // ── 7. Save ──
  SAVE {X_train_scaled, y_train_scaled,
        X_val_scaled,   y_val_scaled,
        X_test_scaled,  y_test_scaled} TO scaled_data.npz

  SAVE scaler_X TO scaler_X.save
  SAVE scaler_y TO scaler_y.save

  OUTPUT: scaled_data.npz, scaler_X.save, scaler_y.save

END DatasetScaling
```

BIOGRAPHY

Cindy Hapsari was born in Bekasi on August 6, 2004. She is an Indonesian citizen currently domiciled in Bekasi and residing in Buleleng or Jakarta. She completed her elementary education at SDN Harapan Baru 4 Bekasi (2016), junior high school at SMPN 21 Bekasi (2019), and senior high school at SMAN 14 Bekasi, majoring in Mathematics and Natural Sciences (2022). In the same year, she enrolled in the Bachelor's program in Informatics Engineering at Ganesha University of Education, specializing in artificial intelligence, particularly data mining, time-series forecasting, computer vision, and natural language processing.

During her studies, she participated in an Independent Study at Dago Engineering, contributed to the Living Lab Smart Fisheries research project on LSTM-based IoT aquaculture systems, and completed an National Internship ‘Magang Berdampak’ at Vidio (EMTEK Group) as an AI Content Insight Intern, developing a production-ready predictive system for film revenue attribution. She was also one of team succeed as finalist of GEMASTIK XVIII in the Internet of Things Division (2025). Her technical competencies include Python, TensorFlow, PyTorch, NLP/LLM, and Data Engineering tools such as Spark, Airflow, SQL, and Streamlit. She aspires to pursue a career as a Data Scientist or AI Engineer focusing on AI-based predictive systems for industrial and environmental applications.