Task 2: AI-Driven IoT Concept

Smart agriculture simulation

Designing a smart agriculture simulation blends real sensor streams with AI predictions to guide irrigation, fertilization, and harvest planning. Tailoring it to Kenyan contexts (e.g., maize, tea, coffee) makes the scenario concrete and impactful for yield stability and resource efficiency.

1. Sensors for crop monitoring and farm operations

Air temperature / humidity (DHT22 or SHT3x) — ambient microclimate.

Soil moisture (capacitance sensor e.g., VH400 or Decagon) — water availability.

Soil temperature (1-wire DS18B20) — root-zone thermal profile.

Soil electrical conductivity (salinity / nutrient proxy) (EC probe) — salinity / fertilizer proxy.

Soil pH (pH electrode + conditioning circuit) — acidity affecting uptake.

Light / PAR (Photosynthetically Active Radiation) (PAR sensor) — usable radiation for photosynthesis.

Solar irradiance / lux (optional) — daylight intensity for growth models.

Rain gauge (tipping-bucket) — precipitation.

Leaf wetness sensor — disease risk modelling.

CO_2 sensor (NDIR) — greenhouse/ambient CO_2 for greenhouse crops

2. AI model to predict crop yields

Objective
Predict total yield (kg/ha or tons/acre) at harvest for each plot/field using sensor time series + weather forecasts + imagery + management metadata (planting date, cultivar, fertilizer, irrigation schedule).

Inputs (features)

a) Time-series sensor-derived features (aggregated by day / week):
   - Daily mean/max/min soil moisture, soil temp, air temp, RH, PAR.
   - Cumulative rainfall since planting.
   - Degree-days (growing degree days).
   - Number of days with leaf-wetness > threshold (disease risk).

b) Management & static features: cultivar, planting date, planting density, fertilizer amounts & dates, irrigation regime, soil type, slope/elevation, GPS location.
c) Exogenous data: historical & forecast weather (temp, rainfall) from meteorological API.
d) Remote-sensing imagery features (optional but very helpful): NDVI time series, canopy cover, early biomass estimates from drone/multispectral.
e) Derived features: rolling means, anomalies vs. historical norms, stress indices (soil moisture deficit), and evapotranspiration estimates (ET0/ETc).

Target
- Continuous: yield per plot (kg/ha) at harvest.
- Optionally: intermediate targets — biomass at stage, harvest date probability.

Model architecture (recommended progressive approach)
Choose models in increasing complexity depending on available data and compute:

Baseline (recommended first step)
Gradient-boosted trees (XGBoost / LightGBM / CatBoost) on aggregated tabular features (daily/weekly aggregated stats + static metadata).
- Why: strong performance on tabular data, fast training, interpretable feature importance, robust to missing values.
- Feature engineering: create seasonal aggregates (first 30/60/90 days), growing degree days, cumulative rainfall, stress counts.
- Evaluation: k-fold time-series split (leave-last-season out), metrics: RMSE, MAE, R^2.

Hybrid model (best if you have imagery)
Image branch (CNN) to extract vegetation indices from images/NDVI → combined with tabular time-series branch (LSTM/CNN) → fused dense layers → yield regression.

- Example: MobileNetV2 (light) for field images → global pooling → concatenate with time-series embedding → fully connected layers.

Uncertainty & explainability

- Use quantile regression or ensemble methods to estimate prediction intervals.
- SHAP values for feature-contribution explanations (works well with tree models).

Training strategy

- Collect multi-season labeled data (3+ seasons ideal).
- Preprocess: align timestamps, fill missing with interpolation + masking flags, and standardize features.
- Create train/validation/test splits by season (train on years 1..N-1, validate on N, test on unseen year). Prevent leakage.
- Hyperparameter tuning: cross-validation on past seasons, early stopping.
- Evaluate on per-plot metrics and aggregated metrics.

Example model hyperparameters (for XGBoost baseline)

- n_estimators: 500 (with early stopping)
- max_depth: 6–10
- learning_rate: 0.01–0.1
- subsample: 0.8
- colsample_bytree: 0.7

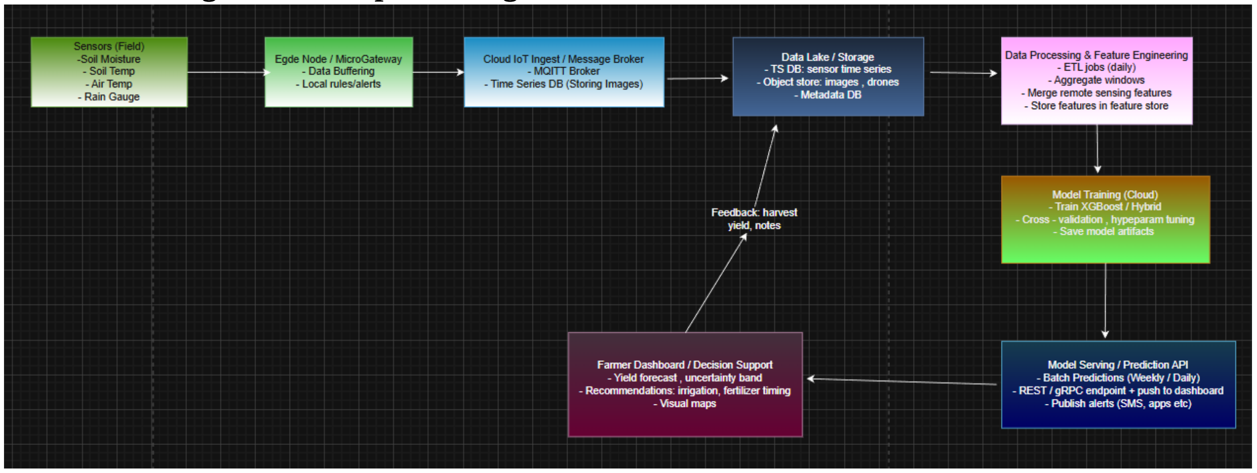
Metrics to report

- RMSE, MAE, R^2, MAPE (if yields not near zero).
- Calibration: predicted vs actual scatter, residual distribution.
- Feature importance and SHAP explanation for top drivers (e.g., cumulative rainfall, GDD, early NDVI).

Deployment considerations

- Edge vs Cloud: Use edge inference for quick local alerts (irrigation deficit warnings). For full retraining and heavy models (CNN + Transformer), use cloud/edge server.
- Model updates: retrain each season or when new labeled harvest data arrives. Implement CI pipeline for data validation + model registry.
- Latency: predictions can be daily/weekly — not real-time strict. Edge inference acceptable for near real-time alerts.

3. Data Flow Diagram for AI processing Data


Sensors (Field)
-Soil Moisture
- Soil Temp
- Air Temp
- Rain Gauge
Egde Node / MicroGateway
- Data Buffering
- Local rules/alerts
Cloud IoT Ingest / Message Broker
- MQITT Broker
- Time Series DB (Storing Images)
Data Lake / Storage
- TS DB: sensor time series
- Object store: images , drones
- Metadata DB
Data Processing & Feature Engineering
- ETL jobs (daily)
- Aggregate windows
- Merge remote sensing features
- Store features in feature store
Model Training (Cloud)
- Train XGBoost / Hybrid
- Cross - validation , hypeparam tuning
- Save model artifacts
Model Serving / Prediction API
- Batch Predictions (Weekly / Daily)
- REST / gRPC endpoint + push to dashboard
- Publish alerts (SMS, apps etc)
Farmer Dashboard / Decision Support
- Yield forecast , uncertainty band
- Recommendations: irrigation, fertilizer timing
- Visual maps
Feedback: harvest
yield, notes

Component descriptions

- **Sensors (field)**: gather raw measurements at configured intervals.
- **Edge Node / Microgateway**: does local aggregation, compression, simple rule-based alerts (e.g., immediate irrigation), and temporary buffering when connectivity is down. Keeps privacy and reduces cloud traffic.
- **Cloud IoT Ingest**: secure ingestion layer; writes timeseries & image objects. Common tools: MQTT, AWS IoT Core, Azure IoT Hub.
- **Data Lake / Time-series DB**: stores raw and processed data for analytics. Use InfluxDB, TimescaleDB, or cloud equivalents (S3 + Athena).
- **Feature Engineering**: ETL jobs to compute aggregated & derived features (daily averages, rolling windows), and to align imagery features.
- **Model Training**: periodic training pipelines (e.g., using Kubeflow, Airflow). Save best models in model registry.
- **Model Serving**: host model endpoint for inference; schedule batch predictions and serve on-demand.
- **Dashboard / Actuators**: visualizations for farmers; actuator commands (irrigation controller) triggered by model or rules.
- **Feedback loop**: harvest yields and farmer corrections flow back to the data lake as labels — essential for continual improvement.

4. Short simulation plan (how to simulate quickly in Colab / locally)

a) **Simulate sensors**: generate synthetic time series (soil moisture dynamics, rainfall events) using simple physical rules (sinusoidal temp, stochastic rainfall).
b) **Make labels**: compute synthetic yield as a function of cumulative water stress, GDD, early NDVI (can be a noisy polynomial).
c) **Train baseline XGBoost** on aggregated features and demonstrate metric improvements as you add more sensors.
d) **Visualize**: time-series plots, feature importances, and predicted vs actual yield.

5. **Risks, limitations & ethical notes.**
   Label quality: yield accuracy depends on reliable harvest measurements.
   Sensor maintenance: fouling, calibration drift → noisy data. Include anomaly detection.
   Connectivity & power: remote areas require resilient power/comms.
   Data privacy: farmer consent for sharing data; secure storage.
   Bias: model trained on one region/soil/cultivar may not generalize — always test on local conditions.