Project Title: Solar-Powered IoT Water-Quality Buoy

---

1. Introduction

A floating, self-powered station that periodically measures key water-quality parameters (pH, turbidity, dissolved oxygen, conductivity, temperature) and transmits data wirelessly to an online dashboard. Designed for long-term deployment in ponds, lakes, or slow-moving rivers.

2. Objectives

- **Continuous Monitoring**: Capture multi-parameter water quality at configurable intervals.
- **Low-Power Operation**: Use solar energy and efficient power management for autonomous operation ≥ 6 months.
- **Long-Range Communication**: Transmit data via LoRaWAN (or NB-IoT) to gateways up to several kilometers away.
- **Outreach & Education**: Provide real-time data visualization for community and school engagement.

3. System Overview

1. **Sensors**: pH probe, optical turbidity sensor, optical dissolved-oxygen (DO) sensor, conductivity probe, DS18B20 temperature sensor.
2. **Controller**: STM32L4 (ARM Cortex-M4) for low-power data acquisition and sensor calibration.
3. **Power**: 5 W solar panel charging a 6 Ah $LiFePO_4$ battery through an MPPT charge controller.
4. **Comms**: LoRaWAN radio module (e.g., RFM95) integrated via SPI; fallback NB-IoT modem if needed.
5. **Enclosure**: IP67-rated waterproof housing with a buoyant float and tether.

4. Hardware Components

Component	Model/Spec	Qty	Estimated Cost
Microcontroller	STM32L476 RG	1	\$15
pH Probe	Analog 0–14 pH probe	1	\$80
Turbidity Sensor	Optical 0–1000 NTU	1	\$50
Dissolved Oxygen Sensor	Optical DO (0–20 mg/L)	1	\$120
Conductivity Probe	0–2000 µS/cm	1	\$40
Temperature Sensor	DS18B20 waterproof module	1	\$5
LoRaWAN Module	RFM95W (915 MHz)	1	\$12
$LiFePO_4$ Battery	6 Ah, 12.8 V	1	\$40
Solar Panel	5 W, 12 V	1	\$20

Component	Model/Spec	Qty	Estimated Cost
MPPT Charge Controller	1 A @ 12 V $LiFePO_4$ support	1	\$15
Waterproof Enclosure	IP67 ABS plastic	1	\$25
Floats & Hardware	PVC, stainless fasteners	1	\$15
Total (est.)			**\$437**

5. Software Architecture

1. **Firmware (STM32)**
2. Sensor drivers (I^2C/SPI/analog) in C.
3. Calibration routines stored in EEPROM.
4. Power state machine: deep-sleep between readings.
5. LoRaWAN stack (e.g., using MCCI LoRaMAC).
6. **Backend & Server**
7. **Go Microservice**: High-performance REST API written in Go (e.g., using Gorilla/Mux or Echo) to ingest LoRaWAN data via MQTT, validate payloads, and write to a time-series database.
8. **MQTT Broker**: e.g., Eclipse Mosquitto for message routing.
9. **GUI Dashboard**
10. **Python Application**: Desktop GUI built with PyQt5 (or Tkinter) for real-time visualization, historical plotting (using matplotlib), and threshold-alert management.
11. **Local Cache**: SQLite for offline storage and quick queries when disconnected.

6. Power Management Power Management

- **Harvesting**: Solar panel → MPPT controller → $LiFePO_4$ battery.
- **Budget**: MCU sleep (5 µA), wake-up → sensors (\~80 mA for 2 s each), LoRa transmit (\~120 mA for 2 s).
- **Estimation**: \~10 mAh per reading cycle; daily readings → \~300 mAh/day; battery + solar balance for multi-month.

7. Communication Strategy

- **Primary**: LoRaWAN (915 MHz) for km-scale, low-power uplinks every 15 min.
- **Fallback**: NB-IoT/2G modem (SIM800C) if LoRa gateway unavailable.
- **Security**: AES-128 encryption (LoRaWAN) and HTTPS/TLS for REST API.

8. Mechanical & Enclosure Design

- **Buoyancy Module**: Two-part foam float housing electronics.
- **Mounting**: Tethered via rope to dock or stake, allowing vertical movement.
- **Sensor Probes**: Mounted via 3D-printed arm, waterproof cable glands, spring-loaded for fixed depth.

9. Data Management & Dashboard

- **Time-Series DB**: InfluxDB for high-resolution data.
- **Visualization**: Grafana or custom React dashboard.
- **Alerts**: Configurable thresholds (e.g., pH < 6.5 or turbidity > 50 NTU) with email/SMS via Twilio.

10. Development Plan & Timeline

Phase	Duration	Deliverables
Requirements & Design	3 weeks	Complete design doc, component sourcing
Hardware Prototype	6 weeks	PCB + enclosure + basic sensors functional
Firmware Development	4 weeks	Sensor drivers, sleep management, LoRa comms
Backend & Dashboard	4 weeks	Data pipeline + web UI deployment
Integration & Testing	3 weeks	Field tests, calibration, power endurance
Outreach Prep	2 weeks	User guide, community/demo materials

11. Testing & Validation

- **Lab Calibration**: Compare sensor readings against known standards.
- **Field Trials**: Deploy two units in different locations for $\geq$ 30 days.
- **Reliability Tests**: Thermal cycling, waterproof soak tests.

12. Outreach & Community Engagement

- Partner with local high school or watershed council.
- Host demo day: live data stream, Q&A, hands-on assembly.
- Publish monthly reports/newsletter with findings.

13. Budget & Resources

- **Estimated Hardware**: \$450 per unit.
- **Cloud Hosting**: \$10–\$20/month (digital-ocean/Heroku).
- **Total**: $\approx$ \$600 for 2 units + 6 months of web hosting.

14. Risks & Mitigation

Risk	Impact	Mitigation
Sensor drift/calibration loss	Data inaccuracy	Regular calibration routines, field checks
Power insufficiency in winter	Downtime	Increase panel size/reading interval

Risk	Impact	Mitigation
Comms failure (no gateway)	Data gaps	Integrate NB-IoT fallback

---

15. UML Responsibilities & Interconnections

```
classDiagram
    class SensorModule {
        // Variables
        +float temperature
        +float pH
        +float turbidity
        +float dissolvedOxygen
        +float conductivity
        +uint32_t timestamp
        // Methods
        +float measureTemperature()
        +float measurepH()
        +float measureTurbidity()
        +float measureDO()
        +float measureConductivity()
    }
    class PowerManager {
        // Variables
        +float batteryLevel
        +float solarVoltage
        +bool isCharging
        +enum PowerState { SLEEP, ACTIVE, CHARGING }
        // Methods
        +void sleep()
        +void wake()
        +void manageSolarCharging()
        +void updateBatteryStatus()
    }
    class CommModule {
        // Variables
        +DataPacket currentPacket
        +string encryptionKey
        +bool lastCommStatus
        +uint32_t lastAttemptTime
        // Methods
        +bool sendData(DataPacket packet)
        +DataPacket encrypt(DataPacket packet)
        +bool fallbackComm(DataPacket packet)
    }
```

```
class Firmware {
    // Variables
    +uint32_t cycleIntervalSec
    +map<string, float> calibrationCoefficients
    +PowerManager::PowerState powerState
    +vector<DataPacket> dataBuffer
    // Methods
    +void runCycle()
    +void calibrateSensors()
    +void powerStateMachine()
    +void enqueueData()
}
class GatewayServer {
    // Variables
    +DBClient dbConnection
    +queue<DataPacket> incomingQueue
    // Methods
    +void receiveData(DataPacket packet)
    +void storeData(DataPacket packet)
    +void forwardToDB()
}
class Dashboard {
    // Variables
    +string apiEndpoint
    +map<string, float> alertThresholds
    // Methods
    +void visualizeData(TimeSeries data)
    +void alertThresholds(DataPacket packet)
    +void fetchLatestData()
}
SensorModule --> Firmware : "sensors → firmware"
PowerManager --> Firmware    : "power status → firmware"
CommModule --> Firmware      : "comm interface → firmware"
Firmware --> CommModule      : "data transmission"
Firmware --> GatewayServer   : "uplink packets"
GatewayServer --> Dashboard  : "API feed → dashboard"
```

End of Design Document.