**Detailed Agile Project Management Plan**

**Project Management Plan: LoRa Spacecraft Telemetry Transmitter**

Version: 1.0

Date: 2025-04-11

Project Start Date (Illustrative): 2025-04-14

Target Hardware: LilyGO T-Beam S3 Supreme (SX1262 Version)

**1. Project Overview**

* **1.1. Purpose:** To develop, test, and document firmware for the LilyGO T-Beam S3 Supreme board, enabling it to function as a reliable LoRa-based telemetry transmitter. The system will collect data from onboard sensors, package it (inspired by CCSDS), apply error control (CRC & Reed-Solomon FEC), and transmit it periodically.
* **1.2. Goals:**
  + Successfully initialize and read data from all specified onboard sensors (BME280, QMI8658, QMC6310, GPS, AXP2101 PMU).
  + Implement a defined data packet structure (CCSDSPacket).
  + Implement CRC-16 calculation for data integrity.
  + Implement Reed-Solomon RS(255, 223) encoding for Forward Error Correction.
  + reliably transmit encoded data packets via the SX1262 LoRa module at configured intervals.
  + Display relevant sensor data and system status on the onboard OLED display.
  + Ensure stable operation on the target hardware.
* **1.3. Scope:**
  + **In Scope:** Firmware development for the transmitter device based on the provided code structure (TelemetryTx.ino, boards.cpp, LoRaBoards.h); Integration and testing of specified sensors; Implementation of packetization, CRC, RS-FEC; LoRa transmission logic; Display interface logic; Board-specific initialization including AXP2101 power management; Basic error handling (init failures, display); Documentation (README, code comments).
  + **Out of Scope:** Development of a LoRa *receiver* station; Advanced low-power optimization (beyond basic PMU rail control); Cloud integration or data backend; Over-the-air firmware updates; Comprehensive hardware design/modification; Formal environmental testing (temperature, vibration etc.); Web interface or mobile application.
* **1.4. Success Criteria:**
  + The firmware compiles, uploads, and runs stably on the T-Beam S3 Supreme.
  + Data from all integrated sensors is read and correctly represented in the telemetry packet.
  + Packets are correctly formatted, including CRC and RS-FEC encoding.
  + Packets are successfully transmitted via LoRa at the intended frequency and parameters.
  + The OLED display cycles through screens showing accurate, real-time data.
  + The system meets basic functional requirements outlined in the Product Backlog.

**2. Project Team & Roles (Scrum)**

* **2.1. Product Owner (PO):** Responsible for defining features, maintaining and prioritizing the Product Backlog, representing stakeholder interests, and approving completed work. (Placeholder: *Project Lead/Stakeholder*)
* **2.2. Scrum Master (SM):** Responsible for facilitating the Scrum process, removing impediments for the Development Team, coaching the team on Agile principles, and ensuring adherence to Scrum practices. (Placeholder: *Team Lead or Designated Developer*)
* **2.3. Development Team:** Cross-functional group responsible for developing, testing, and delivering the firmware increments. Possesses skills in embedded C++, Arduino, hardware interfacing, LoRa, sensor integration, testing, and debugging. (Placeholder: *1-3 Embedded Developers*)

**3. Methodology**

* **3.1. Framework:** Scrum.
* **3.2. Sprint Length:** **2 weeks**. This duration allows for meaningful development progress while accommodating the time needed for hardware testing, integration, and debugging cycles.
* **3.3. Scrum Ceremonies:**
  + **Sprint Planning:** (Beginning of Sprint, ~2-4 hours) Select high-priority items from the Product Backlog to form the Sprint Backlog. Define a Sprint Goal. Break down items into tasks.
  + **Daily Scrum:** (Daily, 15 minutes) Quick synchronization meeting: What did I do yesterday? What will I do today? Any impediments?
  + **Sprint Review:** (End of Sprint, ~1-2 hours) Demonstrate the working increment developed during the Sprint to the Product Owner and stakeholders. Gather feedback.
  + **Sprint Retrospective:** (End of Sprint, ~1 hour) Team reflects on the past Sprint: What went well? What could be improved? What actions to take next Sprint?
* **3.4. Tools:**
  + **Backlog Management:** Jira, Trello, Azure DevOps Boards, or a physical board.
  + **Version Control:** Git (e.g., GitHub, GitLab, Bitbucket).
  + **Communication:** Slack, Microsoft Teams, Email.
  + **Documentation:** Confluence, Wiki, Markdown files in Git repo.
* **3.5. Flexibility Note:** While Sprints provide structure, hardware testing and debugging may require focused effort that can span Sprint boundaries or necessitate adjustments to Sprint scope during planning. Continuous integration testing on hardware is encouraged.

**4. Product Backlog (Initial - High Level & Prioritized)**

*(This is an illustrative starting point based on the provided code; the PO will refine and reprioritize)*

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority** | **Epic** | **User Story / Task** | **Notes** |
| 1 | Board Bring-up | Setup basic Arduino project structure, Initialize Serial | Foundation |
| 2 | Board Bring-up | Implement basic setupBoards() & LoRaBoards.h structure | HAL base |
| 3 | PMU | Implement beginPower() for AXP2101 detection & basic init | Critical for power |
| 4 | PMU | Configure & Verify core power rails (3.3V for ESP32, LoRa, Sensors) via beginPower() | Essential for peripherals |
| 5 | Board Bring-up | Verify I2C Scan (scanDevices) identifies PMU & Display | Hardware check |
| 6 | LoRa Communication | Initialize SX1262 using RadioLib, configure basic LoRa parameters | Radio setup |
| 7 | LoRa Communication | Implement basic LoRa Transmit (radio.startTransmit) of a fixed test payload | Test Tx path |
| 8 | LoRa Communication | Implement TxDone callback (setFlag) mechanism | Async handling |
| 9 | BME280 Sensor | Integrate Adafruit BME280 library, Read Temp/Press/Hum (updateBME280Data) | Environment sensing |
| 10 | IMU/Mag Sensors | Integrate QMI8658 & QMC6310 custom libraries | Requires libraries |
| 11 | IMU/Mag Sensors | Read raw Accel/Gyro data (updateQMI8658Data) | Inertial sensing |
| 12 | IMU/Mag Sensors | Read raw Magnetometer data (updateQMC6310Data) | Magnetic sensing |
| 13 | Data Handling | Define CCSDSPacket struct in TelemetryTx.ino | Data structure |
| 14 | Data Handling | Implement buildTelemetryPacket to populate basic fields (ID, Time) | Packet assembly base |
| 15 | Data Handling | Populate CCSDSPacket with live BME280 & IMU/Mag data in buildTelemetryPacket | Integrate sensor data |
| 16 | Error Control | Implement calculateCRC() function | CRC logic |
| 17 | Error Control | Add CRC calculation & insertion into buildTelemetryPacket | Integrity check |
| 18 | LoRa Communication | Integrate transmit cycle into loop() based on transmittedFlag | Continuous operation |
| 19 | Error Control | Integrate RS-FEC.h library | Requires library |
| 20 | Error Control | Add RS(255, 223) encoding (rs.Encode) to buildTelemetryPacket / transmit buffer | Forward Error Correction |
| 21 | GPS | Initialize SerialGPS & Integrate TinyGPS++ library | GPS setup |
| 22 | GPS | Implement GPS parsing loop in loop() (gps.encode) | Read GPS data |
| 23 | GPS | Implement L76K/UBlox detection logic in beginGPS | Auto-detect module |
| 24 | GPS | Populate CCSDSPacket with GPS data (Lat, Lon, Alt, Time, Sats, etc.) | Integrate GPS data |
| 25 | Display | Implement beginDisplay() using U8g2 library | Display init |
| 26 | Display | Implement screen cycling logic (updateDisplay) | Screen management |
| 27 | Display | Implement data rendering screens (drawDisplay, drawGPS...) for all sensor data | User feedback |
| 28 | PMU | Implement PMU data reading (Batt V/%, Charging) (updatePMUData) & populate packet | Battery monitoring |
| 29 | IMU/Mag Sensors | Implement heading calculation using atan2 & configurable declination | Orientation |
| 30 | PMU | Implement PMU interrupt handling (loopPMU) for button press/charging status | Advanced PMU features |
| 31 | Stability | Implement fatal error handling (displayFatalError) for sensor init failures | Basic robustness |
| 32 | Documentation | Create/Update project README.md | Knowledge sharing |
| 33 | Documentation | Add detailed code comments | Maintainability |
| ... | Bug Fixing/Refinement | Address issues identified during testing, optimize performance | Quality |

**5. Sprint Planning (Illustrative Initial Sprints)**

* **Sprint 1 Goal:** Establish basic board operation and power foundation.
  + *Tasks:* Setup project, basic setupBoards/LoRaBoards.h, Serial output verified, beginPower detects AXP2101, core 3.3V rails verified with multimeter.
* **Sprint 2 Goal:** Integrate LoRa transmit basics & environmental sensor.
  + *Tasks:* Configure SX1262, transmit fixed test packet, implement TxDone callback, integrate BME280, read/print Temp/Press/Hum data.
* **Sprint 3 Goal:** Integrate IMU/Mag sensors & define data packet structure.
  + *Tasks:* Integrate QMI/QMC libraries, read raw Accel/Gyro/Mag data, define CCSDSPacket struct, implement basic buildTelemetryPacket.
* **Sprint 4 Goal:** Integrate sensor data into packet & implement CRC.
  + *Tasks:* Populate CCSDSPacket with BME/IMU/Mag data, implement calculateCRC, add CRC to packet, modify LoRa Tx to send partial packet.
* **Sprint 5 Goal:** Implement RS-FEC Encoding & GPS Reading.
  + *Tasks:* Integrate RS-FEC library, add encoding step, initialize SerialGPS/TinyGPS++, parse NMEA data, populate packet with basic GPS fields.
* **Sprint 6 Goal:** Implement Display Interface.
  + *Tasks:* Initialize U8g2, implement screen cycling, render BME/IMU/LoRa status screens.
* *Subsequent Sprints:* Add remaining display screens, GPS detection logic, PMU readings/interrupts, heading calculation, stability testing, documentation, bug fixing.

**6. Hardware & Environment**

* **Hardware:**
  + LilyGO T-Beam S3 Supreme (SX1262 Version) - *Multiple units recommended for testing/spares.*
  + LiPo Battery (compatible with T-Beam JST connector).
  + USB-C Cable.
  + 5V USB Power Supply / Computer USB Port.
  + (Optional but Recommended) LoRa Receiver (for testing), Multimeter, Oscilloscope/Logic Analyzer, Bench Power Supply, GPS Antenna (if needed).
* **Software:**
  + Arduino IDE (latest version).
  + ESP32 Board Support Package (latest stable version).
  + Required Arduino Libraries (see Dependencies section).
  + Serial Monitor application (IDE built-in or external like PuTTY/minicom).
  + Git client.
  + Project Management Tool (Jira, Trello, etc.).
* **Testing Environment:**
  + Development PC.
  + Dedicated test bench with T-Beam, power supply, connections for debugging.
  + Area suitable for LoRa range testing (potentially outdoors).
  + Area with GPS satellite visibility.

**7. Risk Management**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk Category** | **Risk Description** | **Likelihood** | **Impact** | **Mitigation Strategy** |
| **Hardware** | T-Beam board DOA or malfunction during development | Low-Med | High | Order spare units; Perform initial hardware self-tests; Handle board carefully (ESD). |
| **Hardware** | Sensor (IMU/Mag/BME/GPS) integration issues/failure | Medium | Med | Test sensors individually first; Verify connections/pinouts; Use logic analyzer; Consult datasheets; Source spares. |
| **Hardware** | AXP2101 PMU configuration issues (incorrect rails, stability) | Medium | High | Double-check schematic vs boards.cpp; Test power rails meticulously; Use bench supply for controlled tests. |
| **Software/Library** | Bugs or conflicts in required libraries | Medium | Med | Use stable library versions; Test library updates carefully; Isolate issues; Check library issue trackers; Consider alternatives. |
| **Software/Integration** | Difficulty integrating RS-FEC or custom sensor libraries | Medium | Med | Start with library examples; Test in isolation; Step-through debugging; Ensure correct library installation. |
| **Communication** | LoRa range/reliability lower than expected | Medium | Med | Test in target environment; Optimize LoRa parameters (SF, Power); Check antenna/connector; Use better antennas. |
| **Communication** | GPS slow/no fix | Medium | Low-Med | Test outdoors with clear sky view; Use active antenna if needed; Allow sufficient fix time; Check antenna connection. |
| **Project Management** | Scope creep / Uncontrolled feature addition | Medium | Med | Strict backlog grooming by PO; Clear definition of MVP/Releases; Say "no" or defer features. |
| **Project Management** | Hardware debugging delays impact Sprint commitments | Medium | Med | Allocate buffer time; Allow flexibility in Sprint scope; Use Kanban-like flow for blocking hardware issues. |

**8. Definition of Done (DoD)**

A Product Backlog Item (User Story/Task) is considered "Done" when it meets all the following criteria:

* Code successfully compiles without errors or critical warnings.
* Code implements all acceptance criteria defined for the item.
* Code is formatted according to agreed style guidelines.
* Code includes relevant comments explaining complex sections.
* Code has been peer-reviewed (if team size > 1).
* Unit tests pass (if applicable).
* Integration tests pass (if applicable).
* **Functionality is demonstrated working correctly on the target T-Beam S3 Supreme hardware.**
* Performance meets requirements (e.g., timing constraints).
* Relevant documentation (README, comments, wiki) is updated.
* Code is successfully merged into the main development branch in the Git repository.

**9. Release Planning (High-Level Milestones)**

* **Milestone 1 (Core Tx Functional):** Basic board bring-up, PMU power, core sensors (BME/IMU/Mag) reading, basic packet structure, CRC, LoRa transmission working.
* **Milestone 2 (Full Data Acquisition & FEC):** GPS integration working, RS-FEC encoding implemented, full CCSDSPacket populated and transmitted.
* **Milestone 3 (Display & PMU Integration):** Display interface complete showing all data, PMU data reading & interrupts handled, heading calculation refined.
* **Release 1.0 (Stable):** All features implemented, stability testing passed, documentation complete.

**10. Communication Plan**

* **Meetings:** Daily Scrum, Sprint Planning, Sprint Review, Sprint Retrospective, Ad-hoc technical discussions, Backlog Refinement sessions.
* **Tools:**
  + *Instant Messaging:* Slack/Teams for quick questions and coordination.
  + *Task Tracking:* Jira/Trello/Azure DevOps for Product/Sprint Backlogs.
  + *Documentation:* Confluence/Wiki/Git Repo (Markdown files) for persistent information.
  + *Code Repository:* Git (GitHub/GitLab/etc.) for version control and code reviews.
* **Reporting:** Sprint Burndown charts, Task Board status, Sprint Review demos.

**11. Budget / Resources**

* **Hardware:** Cost of T-Beam S3 Supreme boards, batteries, antennas, potentially test equipment.
* **Software:** Arduino IDE and libraries are typically free/open-source. Project management tools may have costs.
* **Personnel:** Developer time is the primary resource cost.

**12. Document Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** |
| 1.0 | 2025-04-11 | Tanay C. | Initial draft based on provided code analysis |