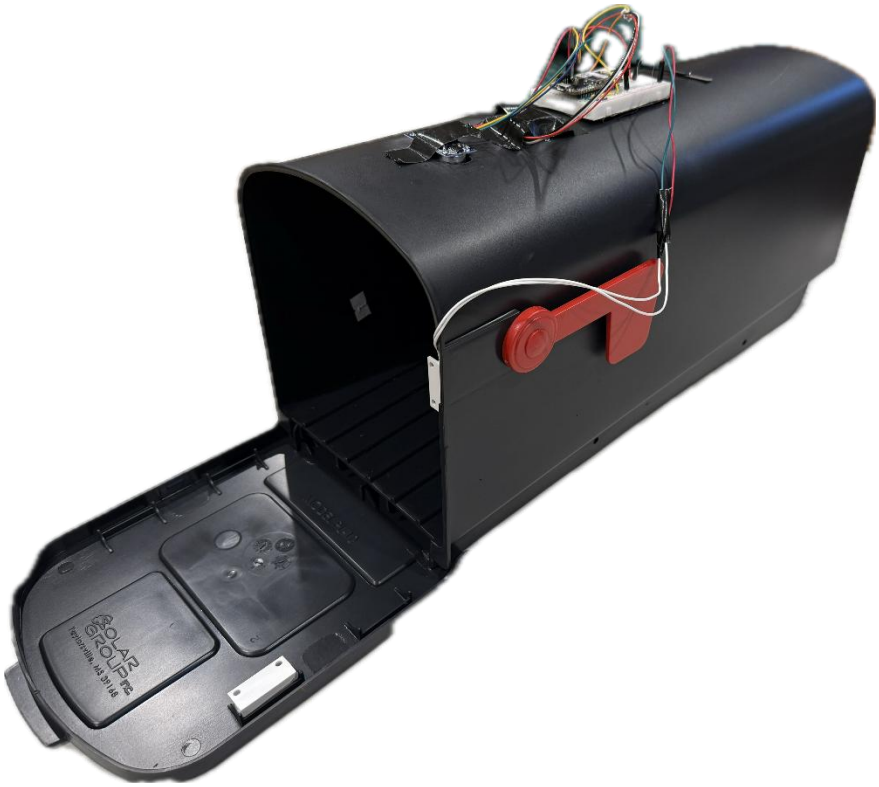
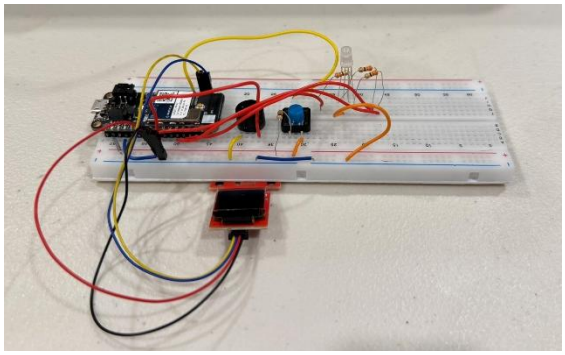
University of Southern California TAC 348 - Making Smart Devices: Introduction to Electronics/Wearables Fall 2025

Final Project: Smart Box

DEVELOPER DOCUMENTATION





Henry Glover

haglover@usc.edu

Contents

Overview

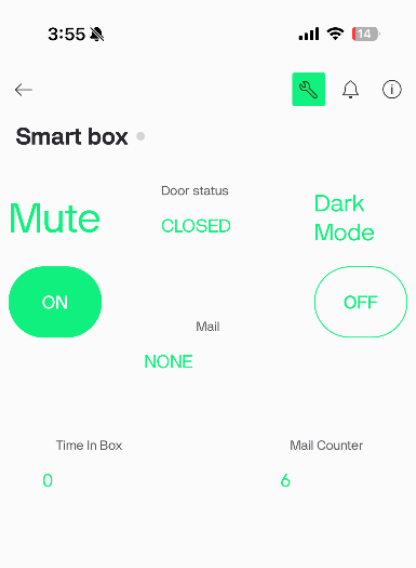
Smart Mailbox Notification System

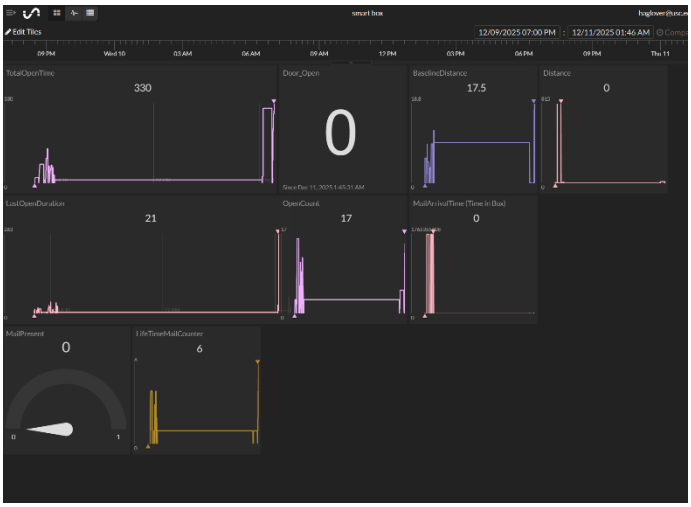
Mail delivery at my residence occurs at inconsistent times, often requiring multiple manual checks throughout the day with no guarantee that mail has arrived. This process is inefficient, inconvenient, and sometimes affected by weather, distance, or schedule constraints. The Smart Box solves this problem by automatically detecting when the mailbox door is opened and whether new mail has been deposited, then notifying the user in real time through both visual indicators and digital updates. In addition to eliminating unnecessary trips to the mailbox, the system also stores historical delivery data, allowing users to track trends and organize their mail-related activity more efficiently.

Target Audience

This system is designed for homeowners, renters, and small office environments seeking a simple and reliable way to know exactly when mail is delivered. It is especially beneficial for individuals who receive time-sensitive documents or packages, those who have mailboxes located far from their residence, or anyone with a P.O. box or detached drop-off location. The system also appeals to smart-home enthusiasts who want a functional and modern device that captures long-term delivery history and provides insightful usage analytics.

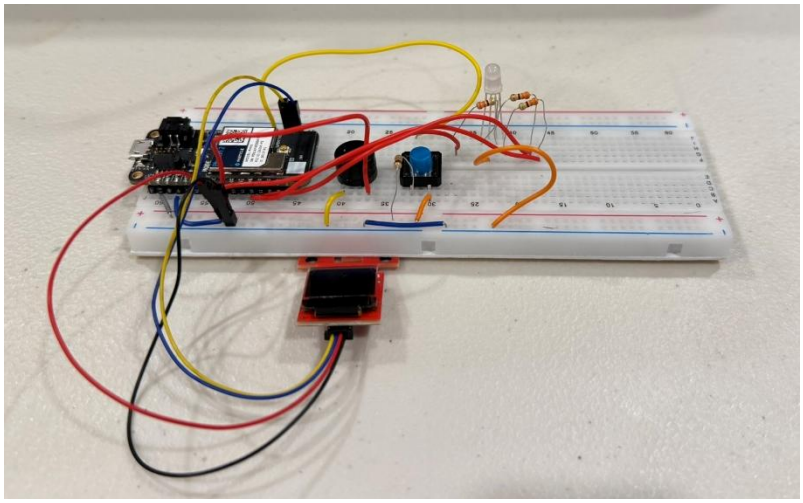
Users can use the Initial State web dashboard, the Blynk app, or the included desk module to track mail delivery and monitor mailbox activity.





Key Features

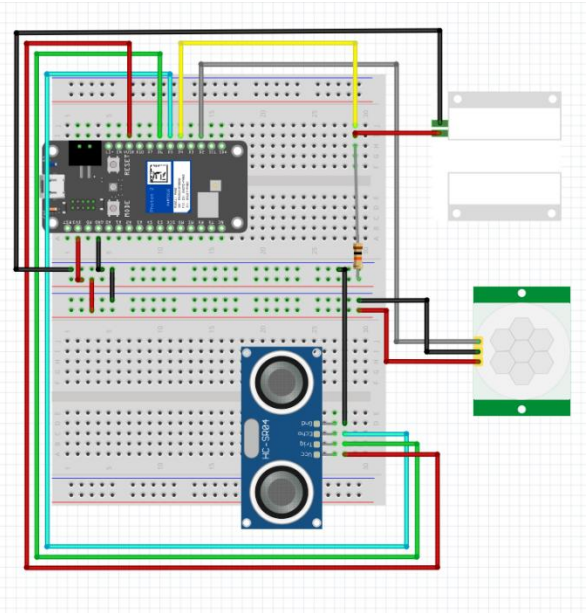
- **Mail Detection**
  A magnetic reed switch monitors mailbox door openings, acting as the initial trigger for potential mail delivery events. An ultrasonic sensor measures distance changes to determine whether physical mail is present inside the mailbox. A PIR motion sensor functions as a redundant confirmation method, detecting movement inside the box to validate that something has been placed inside.
- **Real-Time Notifications**
  Visual alerts on the desk module (RGB LED + OLED display) and digital notifications via the Particle cloud ensure the user knows instantly when mail arrives.
- **Historical Tracking**
  The system logs all delivery events, enabling trend analysis and providing an organized record of mail arrival times.
- **Smart-Home Ready**
  Designed as an add-on smart device that integrates seamlessly into a personal home-technology ecosystem while remaining simple, reliable, and low-maintenance.

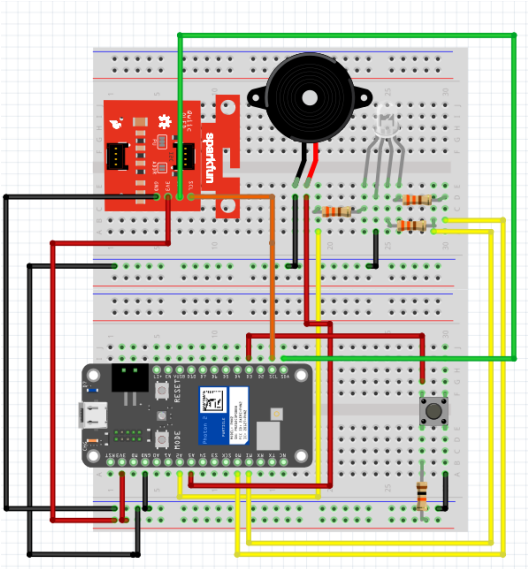


Components on Desk Module

Sensor	Purpose
Button	Reset when mail is retrieved
RGB Led	Visual indicator of mail status
OLED Screen	Visual indicator of mail status
Buzzer	Audio indicator of mail status

Wiring Diagram and Components





The system is composed of two separate components -
1) The sensor module, which is placed inside the mailbox to collect and transmit data.
2) The desk notification module, which sits near the user and provides real-time updates on mailbox status (every second).

The sensor module is powered by a portable power bank, and a LiPo battery. To ensure reliable communication, the sensor module is equipped with an extended-range Wi-Fi module, allowing the system to maintain strong connectivity even at longer distances.

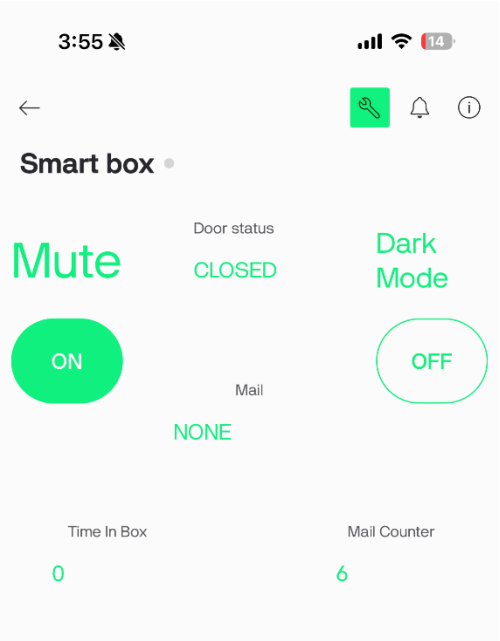
Sensors

The sensors used to achieve the desired functions are shown below

Sensor	Location	Purpose
Magnetic Switch	Mailbox door edge	Detect when the mailbox door opens or closes
Ultrasonic Sensor	Inside top of mailbox	Detect presence or absence of mail by measuring distance to mail surface
Passive Infrared Sensor	Inside top of mailbox	Redundant detection or absence of mail

Device Setup Instructions

1) Wire all the components using the above wiring diagram
    a. Plug device into power connector – device uses cloud for update no need to plug into computer
2) Open Visual Studio Code for both the Sensor module and desk module and configure the project for the Particle Photon Argon 2
    a. Device OS is 6.3.4
3) Cloud flash both projects
4) OLED on the desk module will turn on and say “Smart Box” | Sensor module on mailbox will begin breathing cyan color
5) Open the Blynk app and choose “Smart Box” and log into Initial State and monitor the data streams
6) Use the app to passively monitor your smart mailbox and use the app to configure certain parameters



7) App settings
    a. Can be used to Mute the desk module
    b. Can be used to turn off desk module RGB (dark mode)
    c. Shows Door status, if mail is present, how many times you receive mail (lifetime), and time mail is in the box

Summary of Desk Module Code

Section 1: Libraries, Configuration, and Global Variables

- Includes all required libraries
- Defines Blynk configuration macros
- Declares all pin assignments for the RGB LED, buzzer, button, and OLED reset/DC jumper
- Declares Blynk-related variables (dark mode, mute, and update interval) and timing variables for Blynk updates
- Defines the MailboxState enum
- Tracks the latest sensor data received from the sensor module (door status, PIR motion, distance, timestamps, mailPresent flags, baseline distances, and open/close durations)
- Declares counters (openCount, lifetimeMailCounter) and telemetry publishing control variables for rate-limited per-signal webhooks

Section 2: Setup() Initialization

- Initializes the serial port
- Configures pin modes for the RGB LED, buzzer, and button (INPUT_PULLUP for the button)
- Sets initial outputs
- Initializes the MicroOLED display
- Subscribes to the "mailbox_status" event using Particle.subscribe() so the desk module can receive sensor data from the mailbox module
- Starts the Blynk client using Blynk.begin()
- Draws the initial OLED screen and performs the first Blynk dashboard update

Section 3: Main Loop & High-Level Control

- Runs Blynk.run() to keep the app connection alive and handle incoming virtual writes.
- Calls handleButton(), handleBuzzer(), and handleRgbLed() each loop to manage user input from app
- Calls publishTelemetry() to manage telemetry publishing to Particle webhooks
- Uses a timer (lastBlynkUpdate) to update Blynk every 2 seconds with the latest state (dark mode, mute, mail status, mail time, counter)
- Uses an internal minute-based timer to refresh the OLED display

Section 4: Event Handling from Sensor Module

- mailboxEventHandler() runs whenever a "mailbox_status" event is received
- Parses data in the format: state|distanceCm|doorOpen|pirMotion and updates internal variables
- Tracks door open/close times to compute open durations and total open time
- Uses distance before/after opening, PIR motion, and baseline distance to decide if mail was added or removed
- Updates mailPresent, baselineDistance, and distanceWithMail
- On new mail: plays the “New Mail” song, increments lifetimeMailCounter, and stores mailArrivalTime
- On mail removal: updates mailRemovalTime
- Refreshes previous-state tracking, flags the OLED to update, and calls publishTelemetry()

Section 5: Telemetry Publishing (Webhooks)

- publishSignal(eventName, value) wraps Particle.publish() for individual metrics
- publishTelemetry() runs a timed cycle that publishes one metric per interval to avoid flooding
- Publishes 15 signals in order (state, door, durations, counts, mail flags, timestamps, distance, PIR, baseline, etc.)
- When all signals are sent, the cycle ends and resets for the next update

Section 6: User Input & Feedback Logic

- handleButton() debounces the push button and changes the OLED page when pressed
- handleRgbLed() sets the RGB LED based on system state and Blynk dark mode:
    - Dark mode ON → all LEDs OFF
    - RED → mailbox open too long or sensor error
    - GREEN → mail present
    - OFF → normal / idle
- handleBuzzer() manages all audio alerts:
    - Mute enabled → stops all tones and clears song/warning flags
    - songActive = true → plays the mail chime (playMailSong())

    - MAILBOX_OPEN_TOO_LONG → plays a repeating warning tone (playWarningSound()).

Section 7: OLED Display

- getStatusString() builds a human-readable status based on mailPresent, currentState, and isDoorOpen.
- updateDisplay() draws different pages depending on currentPage:
    - Page 0
        - Title: "Smart Box"
        - Line: "Status: <status string>"
    - Page 1
        - Title: "Smart Box"
        - Door: OPEN/CLOSED
        - PIR: MOTION/NO
        - Dist: <value>cm or N/A

Section 8: Blynk Integration & Widget Handlers

- updateBlynk() sends live values to the app:
    - V0 → Dark Mode (0/1)
    - V1 → Door Status ("OPEN"/"CLOSED")
    - V2 → Mute (0/1)
    - V3 → Mail ("PRESENT"/"NONE")
    - V4 → Mail time in box (seconds)
    - V5 → Lifetime mail counter
- Prints debug messages for mail time and mail count
- BLYNK_WRITE(0) updates blynkDarkMode
- BLYNK_WRITE(2) updates blynkMute and immediately silences the buzzer when mute is enabled

Summary of Sensor Code

Section 1: Libraries, System Configuration, and Global Variables

- Defines all pin assignments for:
    - Ultrasonic sensor (PIN_TRIGGER, PIN_ECHO)
    - PIR motion sensor (PIR_PIN) and indicator LED (PIR_LED_PIN)
    - Magnetic switch (MAG_PIN)
- Declares constants for:
    - Speed of sound in cm, and unit conversion to inches
    - Minimum/maximum valid distance range and warning range
    - Mail-present threshold distance (MAIL_PRESENT_THRESHOLD_IN)
- Declares the MailboxState enum which matches the desk module
- Tracks the current state

Section 2: Setup() Initialization

- Sets pin modes for all hardware:
    - Ultrasonic
    - PIR sensor
    - Magnetic switch

Section 3: Main Loop

- Triggers the ultrasonic sensor and measures echo time with pulseIn()
- Converts echo time to distance in cm and inches
- Reads PIR motion (LOW = motion) and updates the PIR LED
- Reads the magnetic switch to determine if the door is OPEN (HIGH) or CLOSED (LOW)

Section 4: Mailbox State Determination

- Starts each loop with MailboxState newState = IDLE and updates it using sensor readings
- Sets SENSOR_ERROR if distance is invalid or out of range
- Determines mailPresent based on distance compared to MAIL_PRESENT_THRESHOLD_IN
- If the door is open, tracks how long it has been open and sets:
    - MAILBOX_OPEN_TOO_LONG if open > 10 seconds

    - MAIL_PRESENT if mail is detected
    - DOOR_OPEN_NO_MAIL otherwise
- If the door is closed, sets MAIL_PRESENT or IDLE depending on mailPresent
- Updates lastDoorOpen to track open/close transitions.

Section 5: Serial Debug Output

- Prints a status line every loop showing:
    - Ultrasonic distance (in and cm)
    - PIR status (MOTION / NO MOTION)
    - Door status (OPEN / CLOSED)
    - Numeric state value
- Example:
  Ultrasonic: 10.5 in (26.7 cm) | PIR: MOTION | Door: OPEN | State: 2

Section 6: Cloud Publishing to Desk Module

- Sends data to the desk module using Particle.publish("mailbox_status", ...)
- Publishes only on state change or every 2000 ms (whichever comes first).
- Updates currentState and lastPublishMs, then prints [OK] or [FAILED] for each publish
- Ends each loop with delay(500) to control sensor update rate

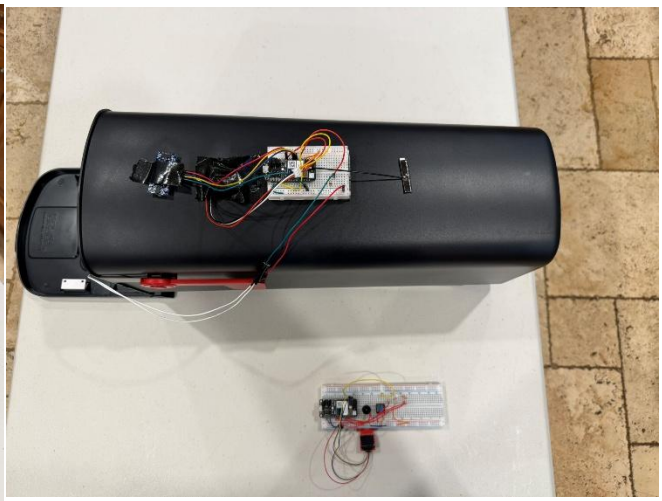
Photon Communication

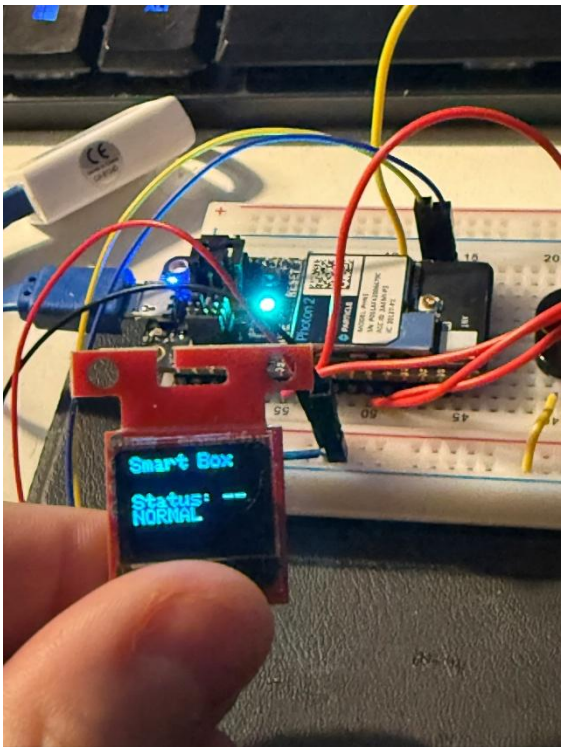
The sensor module and the desk module communicate with each other

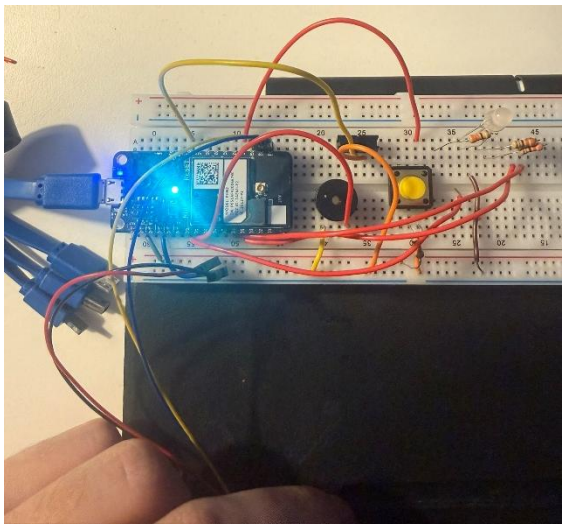
- **Event Name**
    - All sensor data is sent from the Final_Sensor to the desk module using a single event called mailbox_status
- **Publisher** (Sensor Module)
    - The Final_Sensor module calls:
    - Particle.publish ("mailbox_status", data, PRIVATE)
    - String of data is = state, distanceCm, doorOpen, pirMotion
        - MailboxState
            - 0 = IDLE
            - 1 = DOOR_OPEN_NO_MAIL
            - 2 = MAIL_PRESENT
            - 3 = MAILBOX_OPEN_TOO_LONG
            - 4 = SENSOR_ERROR
        - distanceCm = ultrasonic distance in cm (float, 1 decimal)
        - doorOpen = 1 if door is open, 0 if closed
        - pirMotion = 1 if PIR detects motion, 0 otherwise
- **Subscriber** (Desk Module)
    - The desk module subscribes to this event in setup():
    - Particle.subscribe("mailbox_status", mailboxEventHandler, MY_DEVICES);
    - In mailboxEventHandler data is parsed to extract:
        - currentState
        - currentDistance
        - isDoorOpen
        - isPirMotion
    - These values lead to:
        - OLED status pages
        - RGB LED color (mail vs error vs idle)
        - Buzzer alerts (new mail chime, open-too-long warning)

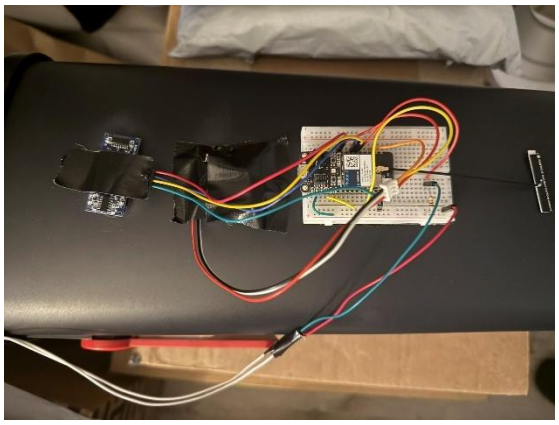
        - Telemetry updates and Blynk updates
    - Final_Sensor module:
        - Publishes only when the mailbox state changes OR every 2 seconds
        - Loop includes a delay(500)
    - Desk module:
        - Reacts immediately on each mailbox_status event
        - Blynk updates (every 2 seconds).

Photos









Datastreams

Search datastream

ID	Name	Pin	Color	Data Type	Units	Is Raw
5	Dark Mode	V0		Integer		false
2	Door Status	V1		String		false
3	Mute	V2		Integer		false
4	Mail	V3		String		false
6	Mail Time In Box	V4		Integer		false
7	Mail counter History	V5		Integer		false

Final_Integration	state	groker.init.st	Enabled	✓ 164 ✕ 62 ↑ 94
Final_Integration	Door_Open	groker.init.st	Enabled	✓ 872 ✕ 199 ↑ 216
Final_Integration	CurrentOpenDuration	groker.init.st	Enabled	✓ 891 ✕ 190 ↑ 207
Final_Integration	LastOpenDuration	groker.init.st	Enabled	✓ 868 ✕ 218 ↑ 285
Final_Integration	TotalOpenTime	groker.init.st	Enabled	✓ 882 ✕ 200 ↑ 211
Final_Integration	OpenCount	groker.init.st	Enabled	✓ 893 ✕ 191 ↑ 205
Final_Integration	MailPresent	groker.init.st	Enabled	✓ 888 ✕ 193 ↑ 210
Final_Integration	LifeTimeMailCounter	groker.init.st	Enabled	✓ 894 ✕ 182 ↑ 196
Final_Integration	MailArrivalTime	groker.init.st	Enabled	✓ 904 ✕ 172 ↑ 161
Final_Integration	MailRemovalTime	groker.init.st	Enabled	✓ 183 ✕ 48 ↑ 48
Final_Integration	LastUpdateTime	groker.init.st	Enabled	✓ 184 ✕ 46 ↑ 38
Final_Integration	Distance	groker.init.st	Enabled	✓ 1K ✕ 242 ↑ 296
Final_Integration	PirMotion	groker.init.st	Enabled	✓ 187 ✕ 50 ↑ 43
Final_Integration	BaselineDistance	groker.init.st	Enabled	✓ 893 ✕ 185 ↑ 184
Final_Integration	DistanceWithMail	groker.init.st	Enabled	✓ 182 ✕ 60 ↑ 63