PieSS - Complete Technical Documentation

Version: 2.0
Last Updated: December 27, 2025
Platform: Raspberry Pi 3 Model B with Raspberry Pi OS (Trixie)

Table of Contents

Project Overview

Purpose

PieSS (Portable ISS) is an autonomous device that tracks the International Space Station and provides visual alerts when it's visible overhead at night. The device features a portable WiFi configuration system allowing it to be used anywhere.

Key Features

- Automatic location detection via IP geolocation
- ISS orbital calculations using Two-Line Element (TLE) data
- Night-time pass filtering based on sunrise/sunset calculations

- Progressive LED countdown alerts (30, 10, 5 minutes)
- Servo-controlled flag raising mechanism
- Real-time directional LEDs showing where to look
- Portable WiFi configuration portal
- Hardware self-test on startup

Operating Modes

Mode 1: ISS Tracker (Normal Operation)

When internet connectivity is available:

- Detects geographic location automatically
- Downloads ISS orbital data from CelesTrak
- Calculates visible night-time passes
- Provides visual alerts and raises flag during passes

Mode 2: WiFi Configuration Portal

When no internet connectivity is available:

- Creates WiFi access point "PieSS-Setup"
- Hosts web configuration interface at 192.168.4.1:8080
- Allows user to scan and connect to WiFi networks
- Automatically switches to ISS Tracker mode once connected

System Architecture

Boot Sequence

```
Power On
   ↓
Raspberry Pi Boot
   ↓
NetworkManager starts
   ↓
boot_decider.service starts
```

```
  ↓
[Check Internet Connectivity]
  ↓
  ├──→ [Has Internet] → Start iss_tracker.service
  │                  └→ ISS Tracker Mode
  │
  └──→ [No Internet] → Configure wlan0 (192.168.4.1)
                     → Start dnsmasq.service
                     → Start hostapd.service
                     → Start wifi_portal.service
                     └→ WiFi Configuration Mode
```

Service Dependencies

```
boot_decider.service
  ├─ Requires: NetworkManager.service
  └─ Starts one of:
     ├─ iss_tracker.service
     │   └─ Requires: pigpiod.service
     │
     └─ WiFi Portal Stack
        ├─ dnsmasq.service
        ├─ hostapd.service
        └─ wifi_portal.service
```

File Structure

```
/home/piess/PieSS/2025/v2/
├── iss_tracker.py          # Main ISS tracking application
├── wifi_portal.py          # WiFi configuration web server
├── boot_decider.sh         # Boot mode decision script
├── hardware_test.py        # Hardware testing utility
├── check_passes.py         # Debug utility for pass calculation
├── venv/                   # Python virtual environment
├── templates/
│   ├── wifi_portal.html    # Main configuration page
│   ├── wifi_result.html    # Connection result page
│   └── logo.png            # PieSS logo
├── stations.tle            # Cached ISS orbital data (auto-generated)
├── de421.bsp               # Sun ephemeris data (auto-downloaded)
└── requirements.txt        # Python dependencies
```

```
/etc/systemd/system/
├── boot_decider.service       # Boot orchestrator
├── iss_tracker.service        # ISS tracker daemon
├── wifi_portal.service        # WiFi portal daemon
└── pigpiod.service            # GPIO daemon

/etc/
├── hostapd/hostapd.conf       # WiFi AP configuration
├── dnsmasq.conf               # DHCP server configuration
└── sudoers.d/piess-sudoers    # Sudo permissions for piess user

/var/log/
└── piess-portal.log           # WiFi portal logs
```

Hardware Setup

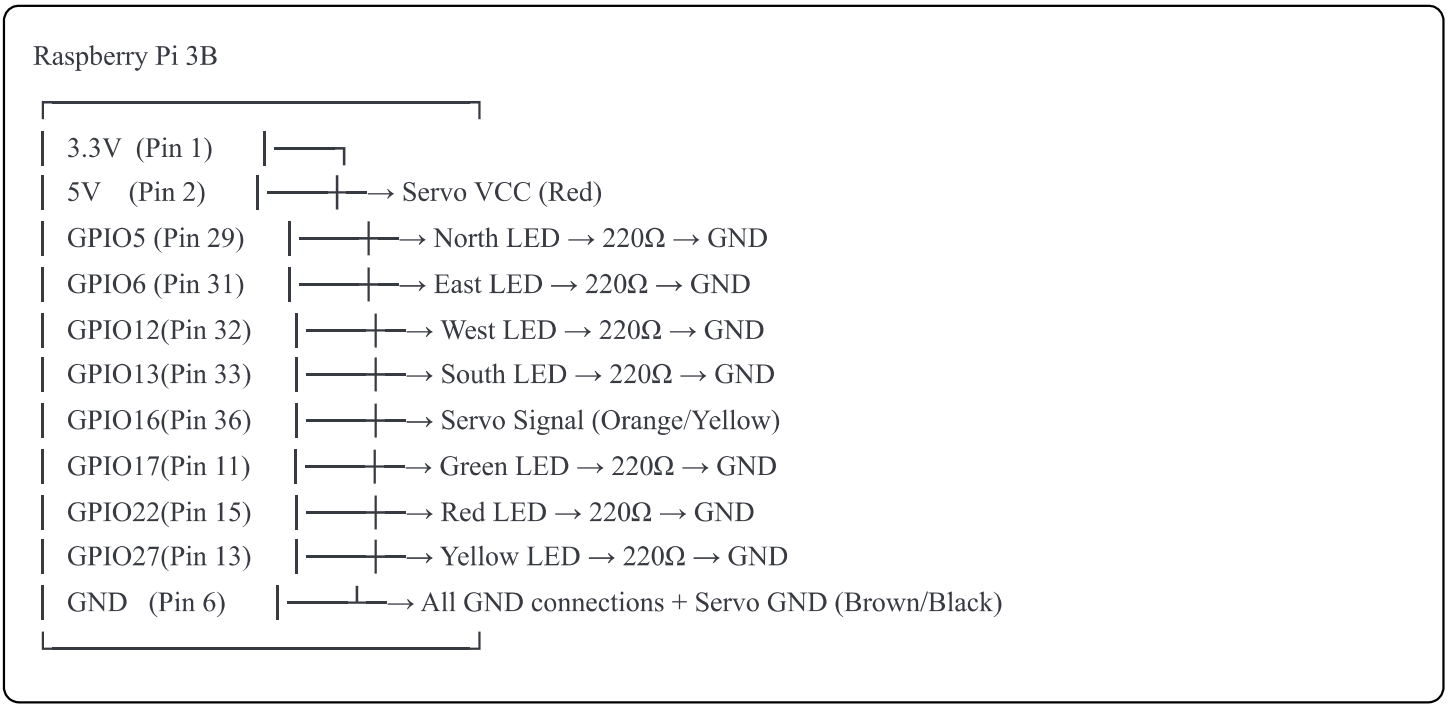
Bill of Materials

Component	Quantity	Specifications	Purpose
Raspberry Pi 3 Model B	1	BCM2837, 1GB RAM	Main controller
MicroSD Card	1	8GB minimum, Class 10	Operating system
Power Supply	1	5V 2.5A USB	Power
Servo Motor	1	5V, 180° rotation	Flag mechanism
Red LED	1	5mm, 20mA	30-minute alert
Yellow LED	1	5mm, 20mA	10-minute alert
Green LED	1	5mm, 20mA	5-minute alert
White/Blue LEDs	4	5mm, 20mA	Directional indicators
Resistors	7	220Ω	LED current limiting
Breadboard	1	Half-size	Prototyping
Jumper Wires	20+	Male-to-female	Connections

GPIO Pin Assignments

GPIO Pin	BCM Number	Component	Function
GPIO 5	BCM 5	North LED	Direction indicator
GPIO 6	BCM 6	East LED	Direction indicator
GPIO 12	BCM 12	West LED	Direction indicator
GPIO 13	BCM 13	South LED	Direction indicator
GPIO 16	BCM 16	Servo PWM	Flag control
GPIO 17	BCM 17	Green LED	5-minute alert
GPIO 22	BCM 22	Red LED	30-minute alert
GPIO 27	BCM 27	Yellow LED	10-minute alert

Wiring Diagram



Servo Configuration

Servo Control:

- PWM Frequency: 50 Hz (20ms period)
- Pulse Width Range: 530-1530 microseconds

- Position UP: 530µs (flag raised)
- Position DOWN: 1530µs (flag lowered)

Servo Connection:

- Brown/Black wire: GND
- Red wire: 5V
- Orange/Yellow wire: GPIO 16 (PWM signal)

LED Configuration

All LEDs use 220Ω current-limiting resistors:

- Forward voltage: ~2V (red), ~2.5V (yellow), ~3V (green/white)
- Forward current: ~15-20mA
- GPIO output: 3.3V

Connection Pattern:

```
GPIO Pin → LED Anode (+) → LED Cathode (-) → 220Ω Resistor → GND
```

Software Installation

Prerequisites

1. **Raspberry Pi OS Installation**
   - Download: Raspberry Pi OS Lite (64-bit recommended)
   - Use Raspberry Pi Imager to flash microSD card
   - Enable SSH in imager settings
   - Configure WiFi credentials (for initial setup)
   - Set hostname: `piess`
   - Create user: `piess` with your password
2. **Boot Configuration**

```
bash
```

```
sudo raspi-config
# Select: 1 System Options → S5 Boot / Auto Login → B1 Console
# This sets the Pi to boot to CLI without auto-login
```

Step-by-Step Installation

1. System Update

bash

```
sudo apt update
sudo apt upgrade -y
```

2. Install System Dependencies

bash

```
sudo apt install -y \
    python3-pip \
    python3-venv \
    pigpio \
    hostapd \
    dnsmasq \
    git \
    nano
```

3. Configure pigpiod

bash

```
# Create pigpiod service
sudo nano /etc/systemd/system/pigpiod.service
```

Paste the following:

ini

```
[Unit]
Description=Pigpio daemon
After=network.target

[Service]
ExecStart=/usr/local/bin/pigpiod -l
PIDFile=/var/run/pigpio.pid
Type=forking
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Enable and start:

```bash
sudo systemctl enable pigpiod
sudo systemctl start pigpiod
```

4. Clone Repository

```bash
cd /home/piess
git clone https://github.com/yourusername/PieSS.git
cd PieSS/2025/v2
```

5. Create Python Virtual Environment

```bash
python3 -m venv venv
source venv/bin/activate
```

6. Install Python Dependencies

```bash
pip install --upgrade pip
pip install flask skyfield requests pigpio gpiozero
```

Create requirements.txt:

```bash
cat > requirements.txt << EOF
flask==3.0.0
skyfield==1.48
requests==2.31.0
pigpio==1.78
gpiozero==2.0.1
EOF
```

7. Configure hostapd

Create configuration:

```bash
sudo nano /etc/hostapd/hostapd.conf
```

Paste:

```ini
interface=wlan0
driver=nl80211
ssid=PieSS-Setup
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=christmas
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

Set permissions:

```bash
sudo chmod 600 /etc/hostapd/hostapd.conf
```

8. Configure dnsmasq

Edit configuration:

```bash
sudo nano /etc/dnsmasq.conf
```

Add these lines:

```ini
interface=wlan0
dhcp-range=192.168.4.10,192.168.4.50,255.255.255.0,12h
domain-needed
bogus-priv
```

9. Configure Sudo Permissions

Create sudoers file:

```bash
sudo visudo -f /etc/sudoers.d/piess-sudoers
```

Paste:

```bash
# Sudoers configuration for PieSS user
# Allows piess user to manage network services without password

piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl start hostapd
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop hostapd
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart hostapd
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl start dnsmasq
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop dnsmasq
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart dnsmasq
piess ALL=(ALL) NOPASSWD: /usr/bin/nmcli
piess ALL=(ALL) NOPASSWD: /usr/sbin/ip
```

Set permissions:

```bash
```

```
sudo chmod 0440 /etc/sudoers.d/piess-sudoers
```

10. Install System Services

boot_decider.service:

```
bash

sudo nano /etc/systemd/system/boot_decider.service
```

```
ini

[Unit]
Description=PieSS Boot Mode Decider
After=network.target NetworkManager.service
Wants=NetworkManager.service

[Service]
Type=oneshot
ExecStart=/home/piess/PieSS/2025/v2/boot_decider.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

iss_tracker.service:

```
bash

sudo nano /etc/systemd/system/iss_tracker.service
```

```
ini
```

```
[Unit]
Description=ISS Tracker
After=network-online.target pigpiod.service
Wants=network-online.target

[Service]
Type=simple
User=piess
WorkingDirectory=/home/piess/PieSS/2025/v2
ExecStart=/home/piess/PieSS/2025/v2/venv/bin/python3 -u /home/piess/PieSS/2025/v2/iss_tracker.py
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

wifi_portal.service:

bash

```
sudo nano /etc/systemd/system/wifi_portal.service
```

ini

```
[Unit]
Description=PieSS WiFi Configuration Portal
After=network.target

[Service]
Type=simple
User=piess
WorkingDirectory=/home/piess/PieSS/2025/v2
ExecStart=/home/piess/PieSS/2025/v2/venv/bin/python3 /home/piess/PieSS/2025/v2/wifi_portal.py
Restart=on-failure
RestartSec=5
StandardOutput=append:/var/log/piess-portal.log
StandardError=append:/var/log/piess-portal.log

[Install]
WantedBy=multi-user.target
```

11. Create Log File

```bash
sudo touch /var/log/piess-portal.log
sudo chown piess:piess /var/log/piess-portal.log
```

12. Make Scripts Executable

```bash
chmod +x /home/piess/PieSS/2025/v2/boot_decider.sh
chmod +x /home/piess/PieSS/2025/v2/iss_tracker.py
chmod +x /home/piess/PieSS/2025/v2/wifi_portal.py
chmod +x /home/piess/PieSS/2025/v2/hardware_test.py
```

13. Enable Services

```bash
# Enable boot decider (this controls everything else)
sudo systemctl enable boot_decider.service

# Enable ISS tracker (will be started by boot_decider when needed)
sudo systemctl enable iss_tracker.service

# Disable AP services (boot_decider controls these)
sudo systemctl disable hostapd
sudo systemctl disable dnsmasq
sudo systemctl disable wifi_portal
```

14. Reload Systemd

```bash
sudo systemctl daemon-reload
```

15. First Boot Test

```bash
sudo reboot
```

System Services

boot_decider.service

Purpose: Orchestrates system startup by determining which mode to operate in based on internet connectivity.

Location: `/etc/systemd/system/boot_decider.service`

Script: `/home/piess/PieSS/2025/v2/boot_decider.sh`

Behavior:

1. Waits 20 seconds for NetworkManager to settle
2. Checks internet connectivity using `nmcli`
3. If internet available: Starts `iss_tracker.service`
4. If no internet: Configures AP mode and starts `wifi_portal.service`

Logs:

```bash
sudo journalctl -u boot_decider.service -f
```

Manual execution:

```bash
sudo /home/piess/PieSS/2025/v2/boot_decider.sh
```

iss_tracker.service

Purpose: Main ISS tracking application that monitors ISS passes and controls hardware.

User: `piess`

Working Directory: `/home/piess/PieSS/2025/v2`

Restart Policy: Always restart on failure

Dependencies:

- Requires internet connectivity
- Requires `pigpiod.service` for GPIO control

Logs:

```bash
sudo journalctl -u iss_tracker.service -f
```

Manual control:

```bash
sudo systemctl start iss_tracker
sudo systemctl stop iss_tracker
sudo systemctl status iss_tracker
```

wifi_portal.service

Purpose: Web-based WiFi configuration interface.

User: `piess`

Port: 8080

Log File: `/var/log/piess-portal.log`

Restart Policy: Restart on failure

Manual control:

```bash
sudo systemctl start wifi_portal
sudo systemctl stop wifi_portal
tail -f /var/log/piess-portal.log
```

pigpiod.service

Purpose: GPIO daemon providing hardware PWM for servo control.

Executable: `/usr/local/bin/pigpiod`

Options: `-l` (enable localhost socket only)

Manual control:

```bash
```

```
sudo systemctl start pigpiod
sudo systemctl status pigpiod
```

hostapd.service

Purpose: Creates WiFi access point when in configuration mode.

Configuration: `/etc/hostapd/hostapd.conf`

Interface: wlan0

SSID: PieSS-Setup

Password: christmas

Manual control:

```
bash

sudo systemctl start hostapd
sudo systemctl status hostapd
sudo journalctl -u hostapd -f
```

dnsmasq.service

Purpose: DHCP and DNS server for WiFi access point.

Configuration: `/etc/dnsmasq.conf`

IP Range: 192.168.4.10 - 192.168.4.50

Lease Time: 12 hours

Manual control:

```
bash

sudo systemctl start dnsmasq
sudo systemctl status dnsmasq
sudo journalctl -u dnsmasq -f
```

Configuration Files

/home/piess/PieSS/2025/v2/boot_decider.sh

bash

```
#!/usr/bin/env bash
# PieSS Boot Mode Decider
# Determines whether to start in normal mode (ISS tracker) or setup mode (WiFi portal)

set -e

if [[ $EUID -ne 0 ]]; then
  echo "[boot_decider] ERROR: must be run as root"
  exit 1
fi

echo "[boot_decider] Starting boot decision process..."

# Give NetworkManager time to initialize and attempt connections
echo "[boot_decider] Waiting for NetworkManager to settle (20s)..."
sleep 20

# Check network connectivity
STATUS=$(nmcli -t -f STATE,CONNECTIVITY general)
echo "[boot_decider] Network status: $STATUS"

# Parse the status
STATE=$(echo "$STATUS" | cut -d: -f1)
CONNECTIVITY=$(echo "$STATUS" | cut -d: -f2)

if [[ "$CONNECTIVITY" == "full" || "$CONNECTIVITY" == "limited" ]]; then
    echo "[boot_decider] Internet available - starting ISS tracker"

    # Ensure AP services are stopped
    systemctl stop hostapd 2>/dev/null || true
    systemctl stop dnsmasq 2>/dev/null || true
    systemctl stop wifi_portal 2>/dev/null || true

    # Start ISS tracker
    systemctl start iss_tracker.service

else
    echo "[boot_decider] No internet - starting WiFi configuration portal"

    # Ensure ISS tracker is stopped
    systemctl stop iss_tracker 2>/dev/null || true

    # Enable WiFi radio
```

```
    nmcli radio wifi on

    # Configure wlan0 for AP mode - MUST happen before starting services
    echo "[boot_decider] Configuring wlan0 interface..."
    ip link set wlan0 down 2>/dev/null || true
    ip addr flush dev wlan0 2>/dev/null || true
    ip addr add 192.168.4.1/24 dev wlan0
    ip link set wlan0 up

    # Verify IP is set
    sleep 1
    WLAN0_IP=$(ip -4 addr show wlan0 | grep -oP '(?<=inet\s)\d+(\.\d+){3}')
    echo "[boot_decider] wlan0 IP address: $WLAN0_IP"

    if [[ "$WLAN0_IP" != "192.168.4.1" ]]; then
        echo "[boot_decider] WARNING: wlan0 IP not set correctly!"
    fi

    # Start dnsmasq first (needs IP to be set)
    echo "[boot_decider] Starting dnsmasq..."
    systemctl start dnsmasq
    sleep 2

    # Then start hostapd
    echo "[boot_decider] Starting hostapd..."
    systemctl start hostapd
    sleep 2

    # Finally start web portal
    echo "[boot_decider] Starting wifi_portal..."
    systemctl start wifi_portal

    echo "[boot_decider] WiFi portal started at http://192.168.4.1:8080"
fi

echo "[boot_decider] Boot decision complete"
```

/etc/hostapd/hostapd.conf

```ini
```

```
# WiFi Access Point Configuration for PieSS
interface=wlan0
driver=nl80211

# Network identification
ssid=PieSS-Setup
hw_mode=g
channel=6

# 802.11n support
ieee80211n=1
wmm_enabled=1

# Authentication
auth_algs=1
ignore_broadcast_ssid=0

# WPA2 encryption
wpa=2
wpa_passphrase=christmas
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

/etc/dnsmasq.conf

```ini
# DHCP Configuration for PieSS Access Point
interface=wlan0
dhcp-range=192.168.4.10,192.168.4.50,255.255.255.0,12h
domain-needed
bogus-priv
```

/etc/sudoers.d/piess-sudoers

```bash
```

```
# Sudoers configuration for PieSS user
# Allows network management without password prompts

piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl start hostapd
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop hostapd
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart hostapd
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl start dnsmasq
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl stop dnsmasq
piess ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart dnsmasq
piess ALL=(ALL) NOPASSWD: /usr/bin/nmcli
piess ALL=(ALL) NOPASSWD: /usr/sbin/ip
```

Application Details

iss_tracker.py

Purpose: Main application for ISS tracking and hardware control.

Key Components:

1. Location Detection

```python
def get_location():
    """Detect geographical location via IP geolocation"""
```

- Tries multiple geolocation providers (ipapi.co, ipinfo.io, ifconfig.co)
- Returns latitude, longitude, elevation
- Falls back to default coordinates if all providers fail
- Default: 43.577090, -79.727520 (Hamilton/Burlington, Ontario area)

2. Satellite Data Management

```python
def get_satellite_data():
    """Load TLE data for ISS, using local cache when available"""
```

- Downloads Two-Line Element (TLE) data from CelesTrak

- Caches data locally in `stations.tle`
- Refreshes every 12 hours
- TLE URL: `https://celestrak.org/NORAD/elements/gp.php?CATNR=25544&FORMAT=tle`

3. Sunrise/Sunset Calculation

```
python

def get_sunrise_sunset(observer_topos, date_t, ephemeris):
    """Calculate sunrise and sunset times for a given date"""
```

- Uses Skyfield's almanac module
- Calculates precise sunrise/sunset for observer's location
- Accounts for timezone and date automatically

4. Night-Time Visibility Check

```
python

def is_visible_at_night(observer_topos, pass_time_t, ephemeris):
    """Check if pass occurs during night time"""
```

- Compares pass time against sunrise/sunset
- Ensures ISS is only tracked when it can be seen
- Accounts for seasonal variations

5. LED Blink Control

```
python

def blink_led(led, duration, blink_rate, check_interval=0.1):
    """Blink an LED for specified duration at given rate"""
```

- Non-blocking LED control
- Precise timing using time.time()
- Configurable blink rates for different alert stages

6. Hardware Self-Test

```python
def test_hardware():
    """Run quick hardware test on startup"""
```

- Tests all 7 LEDs sequentially
- Tests servo up and down movement
- Confirms all GPIO connections working
- Runs automatically on startup

7. Main Tracking Loop

```python
def main():
    """Main ISS tracking loop"""
```

Flow:

1. Detect location
2. Load ephemeris data for sun calculations
3. Reset all hardware to default state
4. Run hardware self-test
5. Enter main loop:
    - Download/refresh ISS orbital data
    - Calculate passes for next 24 hours
    - Filter for night-time passes above 15° elevation
    - Find next visible pass
    - Sleep until 32 minutes before pass
    - Begin progressive LED countdown
    - Raise flag at pass start
    - Show directional LEDs during pass
    - Lower flag after pass completes

- Repeat

Progressive LED Countdown:

Time Remaining	LED	Blink Pattern
30-25 min	Red	4.0s on/off
25-20 min	Red	3.0s on/off
20-15 min	Red	2.0s on/off
15-10 min	Red	1.0s on/off
10-8 min	Yellow	3.0s on/off
8-6 min	Yellow	2.0s on/off
6-5 min	Yellow	1.0s on/off
5-3 min	Green	2.0s on/off
3-1 min	Green	1.0s on/off
1-0 min	Green	0.5s on/off
Pass active	Directional	Solid, updates every 0.5s

Directional LED Logic:

- North: 315° - 45° azimuth
- East: 45° - 135° azimuth
- South: 135° - 225° azimuth
- West: 225° - 315° azimuth
- Updates every 0.5 seconds during pass
- Tracks ISS movement across sky in real-time

Configuration Constants:

```
python
```

```
MIN_ELEVATION = 15.0      # Minimum viewing angle
ALERT_30M = 1800          # 30 minutes in seconds
ALERT_10M = 600           # 10 minutes in seconds
ALERT_5M = 300            # 5 minutes in seconds
TLE_REFRESH_HOURS = 12    # TLE data refresh interval
```

wifi_portal.py

Purpose: Web-based WiFi configuration interface using Flask.

Port: 8080

IP Address: 192.168.4.1 (when in AP mode)

Key Functions:

1. Network Scanning

```
python

def scan_networks():
    """Scan for WiFi networks, temporarily stopping AP"""
```

- Stops hostapd and dnsmasq
- Runs `nmcli dev wifi rescan`
- Parses network list
- Restarts AP services
- Returns list of available networks with signal strength and security

2. AP Mode Control

```
python

def stop_ap_mode():
    """Temporarily stop AP mode to allow WiFi scanning"""

def start_ap_mode():
    """Restart AP mode"""
```

- Manages hostapd and dnsmasq services
- Ensures wlan0 has correct IP (192.168.4.1)

- Adds 2-second delays for service stabilization

3. WiFi Connection

```python
def connect_to_network(ssid, password):
    """Connect to WiFi network and shut down AP if successful"""
```

- Uses nmcli to connect
- Verifies connection established
- Keeps AP down if successful
- Restarts AP if connection fails

4. Web Routes

GET /

- Main configuration page
- Shows cached network list (if available)
- Displays configuration form

GET /scan

- AJAX endpoint for network scanning
- Triggers background scan
- Returns JSON response
- Designed to handle disconnection during scan

POST /connect

- Handles WiFi connection request
- Takes SSID and password from form
- Attempts connection
- Shows result page with status

GET /logo.png

- Serves PieSS logo image
- Used by configuration pages

Scan Cache:

```python
scan_cache = {
    'networks': [],        # List of discovered networks
    'timestamp': None,     # When scan was performed
    'scanning': False      # Scan in progress flag
}
```

Templates:

- `wifi_portal.html` - Main configuration interface
- `wifi_result.html` - Connection result page

hardware_test.py

Purpose: Standalone utility to test all hardware connections.

Usage:

```bash
cd /home/piess/PieSS/2025/v2
source venv/bin/activate
sudo systemctl stop iss_tracker
python3 hardware_test.py
# Press Ctrl+C to stop
```

Test Sequence:

1. North LED (0.5s on, 0.2s off)
2. West LED
3. South LED
4. East LED
5. 30-minute LED (red)
6. 10-minute LED (yellow)

7. 5-minute LED (green)
8. Servo UP (2s hold)
9. Servo DOWN (2s hold)
10. 2 second pause, then repeat

Output:

```
=== PieSS Hardware Test ===
Testing all LEDs and servo in sequence...
Press Ctrl+C to exit

--- Test Cycle 1 ---
Testing: North LED (GPIO 5)... OK
Testing: West LED (GPIO 12)... OK
Testing: South LED (GPIO 13)... OK
Testing: East LED (GPIO 6)... OK
Testing: 30-min LED (GPIO 22)... OK
Testing: 10-min LED (GPIO 27)... OK
Testing: 5-min LED (GPIO 17)... OK
Testing: Servo UP (GPIO 16)... OK
Testing: Servo DOWN (GPIO 16)... OK

Waiting 2 seconds before next cycle...

--- Test Cycle 2 ---
...
```

Network Configuration

NetworkManager

PieSS uses NetworkManager for WiFi management. NetworkManager is the default network management system in Raspberry Pi OS Trixie.

Configuration:

- NetworkManager handles WiFi connections automatically
- Saved WiFi networks are stored in `/etc/NetworkManager/system-connections/`
- Boot_decider checks connectivity using `nmcli general status`

Useful Commands:

```bash
# Check network status
nmcli general status

# List WiFi networks
nmcli dev wifi list

# Connect to WiFi
nmcli dev wifi connect "SSID" password "PASSWORD"

# Show saved connections
nmcli connection show

# Delete a connection
nmcli connection delete "SSID"

# Check device status
nmcli dev status
```

Access Point Mode

When PieSS creates an access point:

Interface: wlan0

IP Address: 192.168.4.1/24

DHCP Range: 192.168.4.10 - 192.168.4.50

Network Configuration:

```bash
# Check if wlan0 has correct IP
ip addr show wlan0

# Should show:
# inet 192.168.4.1/24 scope global wlan0

# Check if services are running
systemctl status hostapd
systemctl status dnsmasq
```

Clients connecting will:

1. Associate with "PieSS-Setup" SSID
2. Receive IP via DHCP (192.168.4.10-50 range)
3. Can access web portal at http://192.168.4.1:8080

Firewall Configuration

PieSS does not require special firewall rules. If you have a firewall enabled:

```bash
# Allow incoming on port 8080 (WiFi portal)
sudo ufw allow 8080/tcp

# Allow DHCP
sudo ufw allow 67/udp
sudo ufw allow 68/udp

# Allow DNS
sudo ufw allow 53/tcp
sudo ufw allow 53/udp
```

Troubleshooting

ISS Tracker Issues

Problem: ISS tracker service won't start

Check logs:

```bash
sudo journalctl -u iss_tracker -n 100 --no-pager
```

Common causes:

1. **No internet connectivity**

```bash
```

```
ping -c 3 google.com
```

2. **pigpiod not running**

```
bash

systemctl status pigpiod
sudo systemctl start pigpiod
```

3. **Python environment issues**

```
bash

cd /home/piess/PieSS/2025/v2
source venv/bin/activate
python3 iss_tracker.py  # Run manually to see errors
```

4. **Permission issues**

```
bash

# Check file ownership
ls -la /home/piess/PieSS/2025/v2/
# Should be owned by piess:piess

# Fix if needed
sudo chown -R piess:piess /home/piess/PieSS/2025/v2/
```

Problem: No ISS passes detected

Run debug utility:

```
bash

cd /home/piess/PieSS/2025/v2
source venv/bin/activate
python3 check_passes.py
```

This shows all passes in the next 48 hours. If no passes appear:

- Check location detection is working

- Verify TLE data is being downloaded
- Check MIN_ELEVATION setting (default 15°)

Problem: LEDs not lighting up

Test hardware:

```bash
cd /home/piess/PieSS/2025/v2
source venv/bin/activate
sudo systemctl stop iss_tracker
python3 hardware_test.py
```

Check wiring:

- Verify GPIO pin assignments match code
- Check LED polarity (long leg = anode/positive)
- Verify resistors are in place
- Test LED directly with 3.3V and ground

Check pigpio:

```bash
systemctl status pigpiod
# Should show "active (running)"

# Test GPIO access
sudo pigpiod -v
```

Problem: Servo not moving

Check power:

- Servo needs 5V, not 3.3V
- Verify servo is connected to Pin 2 (5V) and Pin 6 (GND)
- Signal wire to GPIO 16 (Pin 36)

Test servo manually:

```python
import pigpio
pi = pigpio.pi()
pi.set_servo_pulsewidth(16, 1000)  # Mid position
pi.set_servo_pulsewidth(16, 530)   # Up
pi.set_servo_pulsewidth(16, 1530)  # Down
pi.set_servo_pulsewidth(16, 0)     # Off
pi.stop()
```

WiFi Portal Issues

Problem: Access point not appearing

Check services:

```bash
systemctl status hostapd
systemctl status dnsmasq
systemctl status wifi_portal
```

Check wlan0 configuration:

```bash
ip addr show wlan0
# Should show: inet 192.168.4.1/24

# If not, configure manually:
sudo ip addr flush dev wlan0
sudo ip addr add 192.168.4.1/24 dev wlan0
sudo ip link set wlan0 up
```

Check hostapd logs:

```bash
sudo journalctl -u hostapd -n 50
```

Common errors:

- "Could not configure driver mode" → wlan0 is being managed by NetworkManager
- "Failed to create interface" → wlan0 doesn't have IP address

- "Channel X not allowed" → Country code restrictions

Fix NetworkManager conflict:

```bash
sudo nano /etc/NetworkManager/NetworkManager.conf

# Add:
[keyfile]
unmanaged-devices=interface-name:wlan0

sudo systemctl restart NetworkManager
```

Problem: Can't access web portal (192.168.4.1:8080)

Check if portal is running:

```bash
systemctl status wifi_portal
tail -f /var/log/piess-portal.log
```

Check if port is listening:

```bash
sudo netstat -tulpn | grep 8080
# Should show: tcp 0.0.0.0:8080 ... python3
```

Test from Pi itself:

```bash
curl http://192.168.4.1:8080
# Should return HTML
```

Verify client got IP:

- Client should have IP in range 192.168.4.10-50
- Check dnsmasq logs: `sudo journalctl -u dnsmasq -f`

Problem: Network list is empty

Check scan function:

```bash
# Test nmcli directly
nmcli dev wifi list

# Should show available networks
```

Check sudo permissions:

```bash
sudo -l
# Should show nmcli commands as NOPASSWD
```

Check logs when clicking Refresh:

```bash
tail -f /var/log/piess-portal.log
```

Problem: Can't connect to WiFi from portal

Error: "unknown connection"

- WiFi password is incorrect
- SSID doesn't exist or out of range

Error: "Failed to connect"

- Signal too weak
- WPA/WPA2 security mismatch
- DHCP failure on target network

Debug connection:

```bash
```

```
# Try manually
sudo nmcli dev wifi connect "YourSSID" password "YourPassword"

# Check connection status
nmcli con show
nmcli dev status
```

General System Issues

Problem: Service won't start on boot

Check service status:

```bash
systemctl status boot_decider
systemctl status iss_tracker
```

Check for failed services:

```bash
systemctl --failed
```

View boot logs:

```bash
sudo journalctl -b
```

Check service dependencies:

```bash
systemctl list-dependencies boot_decider.service
```

Problem: High CPU usage

Check running processes:

```bash
```

```
top
# Press 'P' to sort by CPU
# Press 'q' to quit
```

Common causes:

- iss_tracker in tight loop (check logs)
- Multiple instances running (check with `ps aux | grep python`)
- LED blinking consuming CPU (expected during countdowns)

Kill stuck process:

```
bash

sudo systemctl stop iss_tracker
sudo pkill -f iss_tracker.py
```

Problem: SD card corruption

Symptoms:

- System won't boot
- Files missing or read-only
- "Read-only file system" errors

Prevention:

- Always shutdown properly: `sudo shutdown -h now`
- Use quality SD card (Class 10, A1 rated)
- Keep SD card clean and cool

Recovery:

```
bash
```

```
# Boot from another SD card
# Mount corrupted card
sudo mount -o remount,rw /dev/sdX1

# Check filesystem
sudo fsck -f /dev/sdX1

# If beyond repair, reflash SD card and restore from backup
```

Maintenance

Regular Maintenance Tasks

Weekly

- Check for system updates: `sudo apt update && sudo apt upgrade`
- Verify ISS tracker is finding passes: `sudo journalctl -u iss_tracker | tail -20`
- Test hardware: `python3 hardware_test.py`

Monthly

- Clean dust from Raspberry Pi and components
- Check LED brightness (may dim over time)
- Verify servo movement is smooth
- Review log files for errors

As Needed

- Update WiFi credentials if changed
- Adjust MIN_ELEVATION if passes not visible
- Replace LEDs if burned out
- Update TLE data manually if stale

Backup Procedures

Backup Configuration Files

```bash
# Create backup directory
mkdir -p ~/piess-backup/$(date +%Y%m%d)

# Backup Python files
cp ~/PieSS/2025/v2/*.py ~/piess-backup/$(date +%Y%m%d)/

# Backup service files
sudo cp /etc/systemd/system/*piess* ~/piess-backup/$(date +%Y%m%d)/
sudo cp /etc/systemd/system/boot_decider.service ~/piess-backup/$(date +%Y%m%d)/

# Backup configuration
sudo cp /etc/hostapd/hostapd.conf ~/piess-backup/$(date +%Y%m%d)/
sudo cp /etc/dnsmasq.conf ~/piess-backup/$(date +%Y%m%d)/
sudo cp /etc/sudoers.d/piess-sudoers ~/piess-backup/$(date +%Y%m%d)/

# Compress backup
cd ~/piess-backup
tar -czf piess-backup-$(date +%Y%m%d).tar.gz $(date +%Y%m%d)/
```

Full SD Card Image

```bash
# From another computer with SD card reader
# (Replace /dev/sdX with your SD card device)
sudo dd if=/dev/sdX of=piess-backup-$(date +%Y%m%d).img bs=4M status=progress

# Compress image
gzip piess-backup-$(date +%Y%m%d).img
```

Log Management

View Recent Logs:

```bash
```

```
# ISS Tracker
sudo journalctl -u iss_tracker -n 100

# WiFi Portal
tail -100 /var/log/piess-portal.log

# Boot Decider
sudo journalctl -u boot_decider -n 50

# All logs since last boot
sudo journalctl -b
```

Clear Old Logs:

bash

```
# Clear systemd journal (keeps last 7 days)
sudo journalctl --vacuum-time=7d

# Clear WiFi portal log
sudo truncate -s 0 /var/log/piess-portal.log
```

Export Logs for Analysis:

bash

```
# Export ISS tracker logs
sudo journalctl -u iss_tracker --since "2025-12-01" > iss-logs.txt

# Export all logs
sudo journalctl --since "2025-12-01" > system-logs.txt
```

Software Updates

Update PieSS Code

bash

```
cd ~/PieSS/2025/v2
git pull origin main

# Restart services
sudo systemctl restart iss_tracker
sudo systemctl restart wifi_portal
```

Update Python Dependencies

```
bash

cd ~/PieSS/2025/v2
source venv/bin/activate
pip install --upgrade pip
pip install --upgrade -r requirements.txt
```

Update System

```
bash

sudo apt update
sudo apt upgrade -y
sudo apt autoremove -y
sudo reboot
```

Technical Reference

Python Dependencies

Core Libraries:

- **Flask** (3.0.0): Web framework for WiFi portal
- **Skyfield** (1.48): Astronomical calculations for ISS tracking
- **Requests** (2.31.0): HTTP client for geolocation and TLE downloads
- **pigpio** (1.78): GPIO control library for servo
- **gpiozero** (2.0.1): LED control library

Installation:

```bash
pip install flask==3.0.0 skyfield==1.48 requests==2.31.0 pigpio==1.78 gpiozero==2.0.1
```

ISS Orbital Mechanics

Two-Line Element (TLE) Format:

```
ISS (ZARYA)
1 25544U 98067A   25360.54166667  .00012345  00000-0  67890-3 0  9999
2 25544  51.6400 123.4567 0001234  12.3456  45.6789 15.54123456789012
```

Key Parameters:

- **Inclination:** ~51.64° (determines ground track)
- **Eccentricity:** ~0.0001 (nearly circular orbit)
- **Period:** ~90 minutes per orbit
- **Altitude:** ~400-420 km above Earth

Visibility Conditions:

1. ISS altitude above horizon > MIN_ELEVATION (15°)
2. Observer in darkness (after sunset, before sunrise)
3. ISS in sunlight (illuminated from above)
4. Clear sky (not checked by PieSS)

Servo Control Theory

PWM (Pulse Width Modulation):

- Frequency: 50 Hz (20ms period)
- Pulse width determines position:
  - 1.0ms (1000µs) = 0°
  - 1.5ms (1500µs) = 90° (center)
  - 2.0ms (2000µs) = 180°

PieSS Servo Settings:

- UP position: 530µs (~-25° from center)
- DOWN position: 1530µs (~95° from center)
- Total travel: ~120°

pigpiod Hardware PWM:

- Uses hardware PWM peripheral
- More accurate than software PWM
- Supports all GPIO pins via pigpio library
- PieSS uses GPIO 16

LED Current Calculations

Ohm's Law: V = I × R

For Red LED:

- GPIO output: 3.3V
- Forward voltage (Vf): 2.0V
- Desired current: 15mA
- Voltage drop across resistor: 3.3V - 2.0V = 1.3V
- Resistor needed: 1.3V / 0.015A = 86.7Ω
- Standard value used: 220Ω (more conservative, ~6mA)

LED Specifications:

Color	Vf (typ)	Current	Resistor
Red	2.0V	15mA	220Ω
Yellow	2.1V	15mA	220Ω
Green	3.0V	15mA	220Ω
White/Blue	3.2V	15mA	220Ω

Using 220Ω for all LEDs:

- Red: (3.3-2.0)/220 = 5.9mA ✓

- Yellow: (3.3-2.1)/220 = 5.5mA ✓
- Green: (3.3-3.0)/220 = 1.4mA ✓ (dimmer, but safe)
- White: (3.3-3.2)/220 = 0.5mA (very dim, may need lower resistor)

Network Protocols

DHCP (Dynamic Host Configuration Protocol):

- Server: dnsmasq on PieSS
- Range: 192.168.4.10 - 192.168.4.50
- Lease time: 12 hours
- Gateway: 192.168.4.1 (PieSS)
- DNS: 192.168.4.1 (forwarded by dnsmasq)

WPA2-PSK (WiFi Security):

- Encryption: AES-CCMP
- Pre-shared key: "christmas"
- More secure than WEP/WPA
- Supported by all modern devices

HTTP (Web Portal):

- Protocol: HTTP (not HTTPS)
- Port: 8080
- Server: Flask development server
- Not secure - only use on trusted networks

Geolocation APIs

PieSS tries these services in order:

1. **ipapi.co**
    - Endpoint: https://ipapi.co/json/
    - Returns: latitude, longitude, city, country
    - Free tier: 1000 requests/day

- No API key required

2. **ipinfo.io**
    - Endpoint: https://ipinfo.io/json
    - Returns: location as "lat,lon" string
    - Free tier: 50,000 requests/month
    - No API key required
3. **ifconfig.co**
    - Endpoint: https://ifconfig.co/json
    - Returns: latitude, longitude
    - Free to use
    - No API key required

Fallback coordinates:

- Latitude: 43.577090
- Longitude: -79.727520
- Location: Hamilton/Burlington, Ontario, Canada

File Formats

TLE (Two-Line Element) Format:

```
Line 0: Satellite name
Line 1: Orbital elements (epoch, drag, etc.)
Line 2: Orbital elements (inclination, eccentricity, etc.)
```

Cached in: `stations.tle`

BSP (Binary Space Partitioning) Ephemeris:

- Format: JPL DE421 ephemeris
- Contains: Sun, Moon, planetary positions
- File: `de421.bsp` (17 MB)
- Auto-downloaded by Skyfield on first run
- Valid: 1900-2050

Service Files (.service):

- Format: systemd unit files
- Sections: [Unit], [Service], [Install]
- Location: `/etc/systemd/system/`

Appendix

Command Reference

System Management

```bash
# Reboot
sudo reboot

# Shutdown
sudo shutdown -h now

# View system info
uname -a
cat /etc/os-release

# Check disk space
df -h

# Check memory
free -h

# Check CPU temperature
vcgencmd measure_temp
```

Service Management

```bash
```

```
# Start service
sudo systemctl start <service>

# Stop service
sudo systemctl stop <service>

# Restart service
sudo systemctl restart <service>

# Enable on boot
sudo systemctl enable <service>

# Disable on boot
sudo systemctl disable <service>

# View status
systemctl status <service>

# View logs
sudo journalctl -u <service> -f
```

Network Management

bash

```
# Check connectivity
ping -c 3 google.com

# WiFi status
nmcli dev wifi

# Connect to WiFi
nmcli dev wifi connect "SSID" password "PASSWORD"

# Show connections
nmcli con show

# Delete connection
nmcli con delete "SSID"

# Check interfaces
ip addr show

# Check routing
ip route show
```

GPIO Management

```bash
# Test pigpiod
systemctl status pigpiod

# GPIO info
gpio readall  # If wiringPi installed

# Test LED
echo 22 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio22/direction
echo 1 > /sys/class/gpio/gpio22/value
echo 0 > /sys/class/gpio/gpio22/value
```

Glossary

AP (Access Point): WiFi router mode where PieSS creates its own network

Azimuth: Horizontal angle measured clockwise from North (0-360°)

BSP: Binary Space Partitioning - ephemeris data format

DHCP: Dynamic Host Configuration Protocol - assigns IP addresses

Elevation: Vertical angle above horizon (0-90°)

Ephemeris: Table of astronomical positions over time

GPIO: General Purpose Input/Output - Raspberry Pi pins

ISS: International Space Station

NORAD: North American Aerospace Defense Command - tracks satellites

PWM: Pulse Width Modulation - method for controlling servos

SSID: Service Set Identifier - WiFi network name

TLE: Two-Line Element - orbital parameter format

UTC: Coordinated Universal Time - standard time reference

WPA2: WiFi Protected Access 2 - encryption standard

Useful Resources

Official Documentation:

- Raspberry Pi: https://www.raspberrypi.com/documentation/
- Skyfield: https://rhodesmill.org/skyfield/
- Flask: https://flask.palletsprojects.com/
- pigpio: https://abyz.me.uk/rpi/pigpio/

ISS Tracking:

- CelesTrak: https://celestrak.org/
- Heavens Above: https://www.heavens-above.com/
- ISS Detector App: https://issdetector.com/

Hardware:

- GPIO Pinout: https://pinout.xyz/
- LED Resistor Calculator: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-led-series-resistor

Troubleshooting Checklist

Before contacting support, verify:

- [ ] Raspberry Pi has power and boots
- [ ] All LEDs physically connected and working (use hardware_test.py)
- [ ] Servo connected and moving (use hardware_test.py)
- [ ] Internet connectivity available
- [ ] pigpiod service running
- [ ] boot_decider service enabled
- [ ] No error messages in journalctl logs
- [ ] SD card has free space (df -h)
- [ ] System is up to date (sudo apt update)
- [ ] Configuration files have correct permissions
- [ ] Virtual environment activated when testing manually

Version History

v2.0 (December 2025)

- Complete rewrite with portable WiFi configuration
- Progressive LED countdown with acceleration
- Sunrise/sunset based night detection
- Hardware self-test on startup
- Improved error handling and logging
- Complete documentation

v1.0 (Initial Release)

- Basic ISS tracking
- Fixed WiFi configuration
- Simple LED alerts
- Manual configuration required

Support

For issues, questions, or contributions:

- GitHub: https://github.com/yourusername/PieSS
- Email: your.email@example.com

When reporting issues, please include:

1. Output of `sudo journalctl -u iss_tracker -n 100`
2. Output of `systemctl status iss_tracker`
3. Output of `python3 --version`
4. Description of expected vs actual behavior
5. Steps to reproduce the issue

Document End

This documentation covers PieSS v2.0 running on Raspberry Pi 3 Model B with Raspberry Pi OS (Trixie). For updates and latest information, refer to the GitHub repository.</parameter>