# PieSS - Complete Technical Documentation

**Platform:** Raspberry Pi 3 Model B with Raspberry Pi OS (Trixie)

**Note:** This documentation covers the current PieSS implementation located in the `2025/v2` directory of the repository. All file paths in this document reference this location ( `/home/piess/PieSS/2025/v2/` ).

---

## 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

### LED Status Indicators

PieSS uses LEDs to indicate system status in addition to ISS tracking:

#### Access Point Mode Indicator

When PieSS cannot connect to WiFi and enters AP configuration mode:

- **Red 30-minute alert LED blinks** (GPIO 22)
- Pattern: 0.5 seconds on, 0.5 seconds off
- Indicates: System needs WiFi configuration
- Automatically stops when successfully connected to WiFi

#### Network Scanning Animation

When scanning for WiFi networks from the configuration portal:

- **Circular LED animation**: North -> East -> South -> West
- Each LED illuminates for 0.125 seconds
- Pattern repeats for duration of scan (~8 seconds)
- Creates a "radar scanning" visual effect
- Indicates: Active network scan in progress

---

## 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
+-- install_piess.sh            # Automated installation script
+-- requirements.txt            # Python dependencies
+-- venv/                       # Python virtual environment
+-- conf/
|   +-- hostapd.conf            # WiFi AP configuration
|   +-- dnsmasq.conf            # DHCP server configuration
|   +-- sudoers_config.sh       # Sudo permissions helper
+-- services/
|   +-- boot_decider.service    # Boot orchestrator
|   +-- iss_tracker.service     # ISS tracker daemon
|   +-- wifi_portal.service     # WiFi portal daemon
|   +-- pigpiod.service         # GPIO daemon
+-- templates/
|   +-- wifi_portal.html        # Main configuration page
|   +-- wifi_result.html        # Connection result page
|   +-- logo.png                # PieSS logo
+-- documentation/
|   +-- PieSS_Full_Documentation.md
|   +-- PieSS_Full_Documentation.pdf
+-- stations.tle                # Cached ISS orbital data (auto-generated)
+-- de421.bsp                   # Sun ephemeris data (auto-downloaded)

/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
+-- piess-install.log           # Installation log
```

## 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 | Color | Function |
|---|---|---|---|---|
| GPIO 5 | BCM 5 | North LED | Green | Direction indicator |
| GPIO 6 | BCM 6 | East LED | Red | Direction indicator |
| GPIO 12 | BCM 12 | West LED | Red | Direction indicator |
| GPIO 13 | BCM 13 | South LED | Yellow | Direction indicator |
| GPIO 16 | BCM 16 | Servo PWM | N/A | Flag control |
| GPIO 17 | BCM 17 | 5-min LED | Green | Alert |
| GPIO 22 | BCM 22 | 30-min LED | Red | Alert |
| GPIO 27 | BCM 27 | 10-min LED | Yellow | Alert |

### Wiring Diagram

```
Raspberry Pi 3B
+-----------------------+
|  5V    (Pin 2)        |---+-> Servo VCC (Red)
|  GPIO5 (Pin 29)       |---+-> North LED -> 220 Ohm -> GND
|  GPIO6 (Pin 31)       |---+-> East LED -> 220 Ohm -> GND
|  GPIO12(Pin 32)       |---+-> West LED -> 220 Ohm -> GND
|  GPIO13(Pin 33)       |---+-> South LED -> 220 Ohm -> GND
|  GPIO16(Pin 36)       |---+-> Servo Signal (Orange/Yellow)
|  GPIO17(Pin 11)       |---+-> Green LED -> 220 Ohm -> GND
|  GPIO22(Pin 15)       |---+-> Red LED -> 220 Ohm -> GND
|  GPIO27(Pin 13)       |---+-> Yellow LED -> 220 Ohm -> GND
|  GND   (Pin 6)        |---+-> All GND connections + Servo GND (Brown/Black)
+-----------------------+
```

### 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 220 Ohm 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 (-) -> 220 Ohm Resistor -> GND
```

---

## Software Installation

### Automated Installation (Recommended)

PieSS includes an automated installation script that handles the entire setup process.

#### Prerequisites

1. **Raspberry Pi OS Installation**
   - Download: Raspberry Pi OS Lite (64-bit recommended)
   - Use Raspberry Pi Imager to flash microSD card
   - **Important settings in Imager:**
     - Enable SSH
     - Set username: `piess`
     - Set password (your choice)
     - Configure WiFi (optional, for initial setup)
     - Set hostname: `piess`
2. **Boot and Connect**

```
# Option 1: SSH using hostname (if mDNS/Bonjour is working)
ssh piess@piess.local

# Option 2: SSH using IP address
# Find the IP address in your router's DHCP client list or connected devices
# Then connect using:
ssh piess@192.168.1.XXX
# Replace XXX with the actual last octet of the IP address
# On Windows, you may need to enable mDNS or use the IP address method
# On macOS/Linux, .local hostnames typically work by default
```

### Installation Steps

```
# 1. Clone the repository
cd ~
git clone https://github.com/CPowerMav/PieSS.git

# 2. Navigate to v2 directory
cd PieSS/2025/v2

# 3. Make installer executable
chmod +x install_piess.sh

# 4. Run the installer
./install_piess.sh

# The installer will:
# - Update system packages
# - Install dependencies (Python, pigpio, hostapd, dnsmasq, NetworkManager)
# - Create Python virtual environment
# - Install Python packages
# - Configure system services
# - Set up sudo permissions
# - Enable services to start on boot
# - Display installation summary

# 5. Reboot the system
sudo reboot
```

### What the Installer Does

The `install_piess.sh` script performs the following tasks:

1. **System Updates**

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

2. **Install Dependencies**
   - Python 3 and pip
   - pigpio (GPIO library)
   - hostapd (WiFi Access Point)
   - dnsmasq (DHCP server)
   - NetworkManager (network management)
   - git (version control)

3. **Python Environment Setup**
   - Creates virtual environment in `venv/`
   - Installs packages from `requirements.txt`:
     - skyfield (satellite tracking)
     - pytz (timezone handling)
     - requests (HTTP requests)

        - flask (web server)
        - pigpio (GPIO control)

4. **Service Configuration**
    - Copies service files to `/etc/systemd/system/`
    - Configures hostapd (WiFi AP settings)
    - Configures dnsmasq (DHCP server)
    - Sets up sudo permissions for the piess user
5. **Service Enablement**
    - Enables boot_decider.service
    - Enables pigpiod.service
    - Does not enable iss_tracker or wifi_portal (boot_decider manages these)

## Manual Installation

If you prefer to install manually or need to troubleshoot:

### 1. Install System Packages

```
# Update package lists
sudo apt update
sudo apt upgrade -y

# Install required packages
sudo apt install -y \
    python3 \
    python3-pip \
    python3-venv \
    pigpio \
    hostapd \
    dnsmasq \
    network-manager \
    git
```

### 2. Clone Repository

```
cd ~
git clone https://github.com/CPowerMav/PieSS.git
cd PieSS/2025/v2
```

### 3. Create Virtual Environment

```
# Create venv
python3 -m venv venv

# Activate venv
source venv/bin/activate

# Install Python packages
pip install -r requirements.txt
```

### 4. Configure Services

```
# Copy service files
sudo cp services/boot_decider.service /etc/systemd/system/
sudo cp services/iss_tracker.service /etc/systemd/system/
sudo cp services/wifi_portal.service /etc/systemd/system/
sudo cp services/pigpiod.service /etc/systemd/system/

# Copy configuration files
sudo cp conf/hostapd.conf /etc/hostapd/hostapd.conf
sudo cp conf/dnsmasq.conf /etc/dnsmasq.conf

# Set up sudo permissions
sudo bash conf/sudoers_config.sh

# Reload systemd
sudo systemctl daemon-reload
```

### 5. Enable Services

```
# Enable boot orchestrator
sudo systemctl enable boot_decider.service

# Enable GPIO daemon
sudo systemctl enable pigpiod.service

# Do NOT enable iss_tracker or wifi_portal
# These are started dynamically by boot_decider
```

### 6. Verify Installation

```
# Check service files
systemctl list-unit-files | grep piess

# Check sudo permissions
sudo -l

# Test virtual environment
source ~/PieSS/2025/v2/venv/bin/activate
python3 -c "import skyfield; print('Skyfield imported successfully')"
```

### 7. Reboot

```
sudo reboot
```

## Python Dependencies

The following Python packages are installed in the virtual environment:

```
skyfield==1.49      # Satellite position calculations
pytz==2024.2        # Timezone handling
requests==2.32.3    # HTTP requests for geolocation
flask==3.0.3        # Web server framework
pigpio==1.78        # GPIO control library
```

## Post-Installation Verification

After reboot, verify the installation:

```
# Check boot_decider status
systemctl status boot_decider

# Check if ISS tracker or WiFi portal started
systemctl status iss_tracker
# OR
systemctl status wifi_portal

# View logs
sudo journalctl -u boot_decider -f
sudo journalctl -u iss_tracker -f
```

---

## System Services

### boot_decider.service

**Purpose:** Boot orchestrator that determines which mode to start based on internet connectivity.

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

**Service File:**

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

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

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

**Script Logic (boot_decider.sh):**

```
#!/bin/bash

LOG_TAG="boot_decider"

log_message() {
    echo "[$LOG_TAG] $1"
    logger -t "$LOG_TAG" "$1"
}

# Wait for NetworkManager to be ready
sleep 5

# Check internet connectivity
if ping -c 3 -W 5 8.8.8.8 &> /dev/null; then
    log_message "Internet detected - starting ISS tracker mode"
    sudo systemctl start iss_tracker.service
```

```
else
    log_message "No internet - starting WiFi configuration mode"

    # Configure wlan0 for AP mode
    sudo ip link set wlan0 down
    sudo ip addr flush dev wlan0
    sudo ip addr add 192.168.4.1/24 dev wlan0
    sudo ip link set wlan0 up

    # Start AP services
    sudo systemctl start dnsmasq.service
    sudo systemctl start hostapd.service

    # Give services time to start
    sleep 3

    # Start web portal with LED indicator
    /home/piess/PieSS/2025/v2/venv/bin/python3 \
        /home/piess/PieSS/2025/v2/blink_ap_led.py &

    sudo systemctl start wifi_portal.service

    log_message "WiFi portal started at 192.168.4.1:8080"
fi
```

**Commands:**

```
# View status
systemctl status boot_decider

# View logs
sudo journalctl -u boot_decider -f

# Manually trigger (for testing)
sudo systemctl start boot_decider
```

### iss_tracker.service

**Purpose:** Main ISS tracking application daemon.

**Location:** `/etc/systemd/system/iss_tracker.service`

**Service File:**

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

[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/iss_tracker.py
Restart=on-failure
RestartSec=10
StandardOutput=journal
StandardError=journal
```

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

**Commands:**

```
# Start tracker
sudo systemctl start iss_tracker

# Stop tracker
sudo systemctl stop iss_tracker

# View status
systemctl status iss_tracker

# View logs
sudo journalctl -u iss_tracker -f

# View recent errors
sudo journalctl -u iss_tracker -p err -n 50
```

### wifi_portal.service

**Purpose:** WiFi configuration web portal daemon.

**Location:** `/etc/systemd/system/wifi_portal.service`

**Service File:**

```
[Unit]
Description=PieSS WiFi Configuration Portal
After=network.target hostapd.service dnsmasq.service
Requires=hostapd.service dnsmasq.service

[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=journal
StandardError=journal
Environment=PYTHONUNBUFFERED=1

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

**Commands:**

```
# Start portal
sudo systemctl start wifi_portal

# Stop portal
sudo systemctl stop wifi_portal

# View status
systemctl status wifi_portal

# View logs
```

```
sudo journalctl -u wifi_portal -f
tail -f /var/log/piess-portal.log
```

### pigpiod.service

**Purpose:** GPIO daemon providing hardware PWM and precise timing.

**Location:** `/etc/systemd/system/pigpiod.service`

**Service File:**

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

[Service]
Type=forking
ExecStart=/usr/bin/pigpiod -l
ExecStop=/bin/systemctl kill pigpiod
Restart=on-failure
RestartSec=5

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

**Commands:**

```
# Start daemon
sudo systemctl start pigpiod

# Stop daemon
sudo systemctl stop pigpiod

# View status
systemctl status pigpiod

# Check if running
ps aux | grep pigpiod
```

## Service Management

### Starting/Stopping Services

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

# Stop a service
sudo systemctl stop <service_name>

# Restart a service
sudo systemctl restart <service_name>

# View status
systemctl status <service_name>
```

### Enabling/Disabling Services

```
# Enable service (start on boot)
sudo systemctl enable <service_name>

# Disable service (don't start on boot)
sudo systemctl disable <service_name>

# Check if enabled
systemctl is-enabled <service_name>
```

### Viewing Logs

```
# Follow live logs
sudo journalctl -u <service_name> -f

# View last 100 lines
sudo journalctl -u <service_name> -n 100

# View logs since boot
sudo journalctl -u <service_name> -b

# View only errors
sudo journalctl -u <service_name> -p err
```

### Service Dependencies

The services have the following dependencies:

```
boot_decider
    └ NetworkManager (waits for network)
    └ Starts one of:
        ├ iss_tracker
        │   └ pigpiod (required)
        │   └ network-online (waits for internet)
        │
        └ WiFi Portal Stack
            ├ hostapd (required)
            ├ dnsmasq (required)
            └ wifi_portal
```

---

## Configuration Files

### /etc/hostapd/hostapd.conf

**Purpose:** WiFi Access Point configuration.

```
# Interface configuration
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
```

**Configuration Details:**

| Parameter | Value | Description |
|---|---|---|
| interface | wlan0 | WiFi interface to use |
| driver | nl80211 | Modern Linux WiFi driver |
| ssid | PieSS-Setup | Network name (visible to users) |
| hw_mode | g | 2.4 GHz band (802.11g) |
| channel | 6 | WiFi channel (1-11) |
| ieee80211n | 1 | Enable 802.11n (faster speeds) |
| wpa | 2 | WPA2 security |
| wpa_passphrase | christmas | Network password |
| wpa_key_mgmt | WPA-PSK | Pre-shared key authentication |
| rsn_pairwise | CCMP | AES encryption |

**Changing Configuration:**

```
# Edit configuration
sudo nano /etc/hostapd/hostapd.conf

# Test configuration
sudo hostapd -dd /etc/hostapd/hostapd.conf

# Restart service
sudo systemctl restart hostapd
```

## /etc/dnsmasq.conf

**Purpose:** DHCP server configuration for Access Point mode.

```
# 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
```

**Configuration Details:**

| Parameter | Value | Description |
|---|---|---|
| interface | wlan0 | Interface to provide DHCP on |

| Parameter | Value | Description |
|---|---|---|
| dhcp-range | 192.168.4.10-50 | IP address pool |
| netmask | 255.255.255.0 | Subnet mask (/24) |
| lease-time | 12h | How long IPs are assigned |
| domain-needed | enabled | Don't forward non-FQDN queries |
| bogus-priv | enabled | Don't forward private IP queries |

**The DHCP Setup:**

- Gateway: 192.168.4.1 (PieSS)
- DNS: 192.168.4.1 (forwarded by dnsmasq)
- Range: 192.168.4.10 - 192.168.4.50
- Subnet: 255.255.255.0
- Max clients: 41

**Testing DHCP:**

```
# Check if dnsmasq is running
systemctl status dnsmasq

# View DHCP leases
cat /var/lib/misc/dnsmasq.leases

# Test DNS forwarding
nslookup google.com 192.168.4.1
```

## /etc/sudoers.d/piess-sudoers

**Purpose:** Grant piess user passwordless sudo for specific commands.

```
# 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
piess ALL=(ALL) NOPASSWD: /usr/bin/kill
piess ALL=(ALL) NOPASSWD: /bin/sh
piess ALL=(ALL) NOPASSWD: /bin/rm
```

**Why These Permissions:**

- `systemctl` commands: Start/stop AP services
- `nmcli` : WiFi network management
- `ip` : Network interface configuration
- `kill` : Stop background processes (LED blinker)
- `sh` : Execute shell commands from Python
- `rm` : Delete network connection profiles

**Security Note:** These permissions are necessary for the WiFi portal to function without manual intervention, but they should only be used on trusted networks.

**Verification:**

```
# Check sudo permissions for piess user
sudo -l -U piess

# Test a command (should not ask for password)
sudo systemctl status hostapd
```

---

## Application Details

### iss_tracker.py

**Purpose:** Main application for ISS tracking and hardware control.

**Key Components:**

#### 1. Location Detection

```
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

```
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

```
def get_sunrise_sunset(observer_topos, date_t, ts, 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

```
def is_visible_at_night(observer_topos, pass_time_t, ts, 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 degrees 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 |

| Time Remaining | LED | Blink Pattern |
|---|---|---|
| 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 degrees - 45 degrees azimuth
- East: 45 degrees - 135 degrees azimuth
- South: 135 degrees - 225 degrees azimuth
- West: 225 degrees - 315 degrees azimuth
- Updates every 0.5 seconds during pass
- Tracks ISS movement across sky in real-time

**Configuration Constants:**

```
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. Includes visual feedback via LED animations.

**Port:** 8080

**IP Address:** 192.168.4.1 (when in AP mode)

**Key Functions:**

**1. Network Scanning with LED Animation**

```
def start_scan_animation():
    """Start LED animation to indicate WiFi scanning in progress"""
```

- Starts circular LED animation (N->E->S->W)
- Each directional LED illuminates for 0.125 seconds
- Creates "radar scanning" visual effect
- Runs in background thread during ~8 second scan
- Automatically stops when scan completes

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

- Triggers LED scanning animation
- 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"""

def stop_ap_led_indicator():
    """Stop the AP mode LED indicator by killing the background process"""
```

- Manages hostapd and dnsmasq services
- Ensures wlan0 has correct IP (192.168.4.1)
- Adds 2-second delays for service stabilization
- Stops blinking LED indicator when connected

#### 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:**

```
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 page
- `wifi_result.html`: Connection result page
- Uses Jinja2 templating engine

**Logging:**

- All activity logged to `/var/log/piess-portal.log`
- Includes timestamps, network operations, errors
- Useful for troubleshooting connection issues

## hardware_test.py

**Purpose:** Standalone hardware testing utility.

**Usage:**

```
# Activate virtual environment
source ~/PieSS/2025/v2/venv/bin/activate

# Run hardware test
python3 hardware_test.py
```

**Test Sequence:**

1. Tests each LED individually (0.5s on)
2. Tests servo up position (2s)
3. Tests servo down position (2s)
4. Confirms all hardware working

**Output:**

```
Starting hardware test...
Testing North LED (GPIO 5)...
Testing East LED (GPIO 6)...
Testing West LED (GPIO 12)...
Testing South LED (GPIO 13)...
Testing Green Alert LED (GPIO 17)...
Testing Red Alert LED (GPIO 22)...
Testing Yellow Alert LED (GPIO 27)...
Testing Servo UP position...
Testing Servo DOWN position...
Hardware test complete!
```

### check_passes.py

**Purpose:** Debug utility for viewing calculated ISS passes.

**Usage:**

```
# Activate virtual environment
source ~/PieSS/2025/v2/venv/bin/activate

# Run pass checker
python3 check_passes.py
```

**Output Example:**

```
Location: 43.5771°N, 79.7275°W
Sunrise: 2025-01-05 07:42:15 EST
Sunset:  2025-01-05 17:18:32 EST

ISS Passes (next 24 hours):
============================

Pass 1:
  Time:    2025-01-05 19:23:45 EST
  Max Alt: 67.3°
  Azimuth: 142.5° (SE)
  Status:  VISIBLE (night-time pass)

Pass 2:
  Time:    2025-01-05 21:01:23 EST
  Max Alt: 23.8°
  Azimuth: 287.1° (W)
  Status:  VISIBLE (night-time pass)

Pass 3:
  Time:    2025-01-06 08:45:12 EST
  Max Alt: 45.2°
  Azimuth: 198.7° (SSW)
  Status:  SKIPPED (daytime pass)
```

**Use Cases:**

- Verify pass calculations
- Check night filtering logic
- Troubleshoot missed passes
- Understand ISS visibility patterns

---

## Network Configuration

### Access Point Details

**Network Name:** PieSS-Setup

**Password:** christmas

**IP Address:** 192.168.4.1

**DHCP Range:** 192.168.4.10 - 192.168.4.50

**Configuration Portal:** http://192.168.4.1:8080

## WiFi Configuration Process

### For Users (Connecting to PieSS)

1. **Power on PieSS** without internet connection
   - Wait 30 seconds for AP mode to start
   - Red LED will blink on the device
2. **Connect to PieSS-Setup network**
   - Open WiFi settings on your phone/computer
   - Look for network "PieSS-Setup"
   - Password: christmas
3. **Open configuration portal**
   - Most devices: Portal opens automatically (captive portal)
   - Manual access: Open browser to http://192.168.4.1:8080
4. **Scan for networks**
   - Click "Scan for WiFi Networks" button
   - Watch circular LED animation during scan
   - Wait for network list to appear (~10 seconds)
5. **Select and connect**
   - Choose your WiFi network from list
   - Enter password
   - Click "Connect"
   - Wait for confirmation (may disconnect briefly)
6. **Verify connection**
   - Check for success message
   - Red LED should stop blinking
   - PieSS will reboot into ISS tracker mode

### Network Transition Flow

```
User Connects to PieSS-Setup
    |
    v
Opens 192.168.4.1:8080
    |
    v
Clicks "Scan for WiFi Networks"
    |
    v
[Portal stops AP temporarily]
    |
    v
[LED scanning animation plays]
    |
    v
[Portal scans for networks]
    |
    v
[Portal restarts AP]
    |
    v
Network list displayed
    |
    v
User selects network & enters password
    |
    v
[Portal attempts connection]
    |
    +---> [Success] ----> AP stops permanently
    |                      LED stops blinking
    |                      System reboots to ISS mode
    |
    +---> [Failure] ----> AP restarts
                         User can try again
```

## NetworkManager Configuration

PieSS uses NetworkManager for WiFi management, which provides several advantages:

**Benefits:**

- Automatic connection to known networks
- Persistent connection profiles
- Better roaming support
- Integration with systemd

**Connection Management:**

```
# List saved connections
nmcli con show

# Delete a connection
nmcli con delete "SSID_NAME"

# Show current connection
nmcli con show --active

# Manually connect
nmcli dev wifi connect "SSID" password "PASSWORD"
```

**Connection Priority:**

NetworkManager automatically connects to networks in this order:

1. Previously successful connections (highest priority)
2. Strongest signal strength
3. Most recent connection

**Troubleshooting Connections:**

```
# Check WiFi status
nmcli radio wifi
```

```
# Turn WiFi on
nmcli radio wifi on

# Rescan for networks
nmcli dev wifi rescan

# Show detailed connection info
nmcli -f ALL con show "CONNECTION_NAME"
```

### Firewall Configuration

PieSS does not use a firewall by default for simplicity, but you can add one:

```
# Install UFW (Uncomplicated Firewall)
sudo apt install ufw

# Allow SSH
sudo ufw allow 22/tcp

# Allow web portal
sudo ufw allow 8080/tcp

# Enable firewall
sudo ufw enable

# Check status
sudo ufw status verbose
```

## Troubleshooting

### Common Issues

#### 1. Service Won't Start

**Symptoms:**

- `systemctl status <service>` shows "failed" or "inactive"
- Errors in journal logs

**Diagnosis:**

```
# Check service status
systemctl status iss_tracker

# View detailed logs
sudo journalctl -u iss_tracker -n 50

# Check if dependencies are running
systemctl status pigpiod
```

**Solutions:**

a) **Missing Python packages:**

```
cd ~/PieSS/2025/v2
source venv/bin/activate
```

```
pip install -r requirements.txt
```

b) **Permission issues:**

```
# Check file ownership
ls -l ~/PieSS/2025/v2/iss_tracker.py

# Fix ownership if needed
sudo chown piess:piess ~/PieSS/2025/v2/*.py
```

c) **pigpiod not running:**

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

### 2. No Internet Connection / Stuck in AP Mode

**Symptoms:**

- PieSS always creates AP, never starts ISS tracker
- Can't connect to home WiFi

**Diagnosis:**

```
# Check connectivity
ping -c 3 8.8.8.8

# Check WiFi status
nmcli dev wifi

# List saved connections
nmcli con show
```

**Solutions:**

a) **Delete bad connection profile:**

```
nmcli con show
nmcli con delete "BAD_CONNECTION_NAME"
```

b) **Manually connect to WiFi:**

```
nmcli dev wifi connect "YOUR_SSID" password "YOUR_PASSWORD"
```

c) **Force network rescan:**

```
sudo systemctl restart NetworkManager
nmcli dev wifi rescan
```

d) **Check router settings:**

- MAC address filtering disabled
- DHCP enabled
- Correct password

### 3. LEDs Not Working

**Symptoms:**

- LEDs don't light up during hardware test
- Some LEDs work, others don't

**Diagnosis:**

```bash
# Run hardware test
cd ~/PieSS/2025/v2
source venv/bin/activate
python3 hardware_test.py
```

**Solutions:**

a) **Check physical connections:**

- LED polarity (long leg = +, short leg = -)
- Resistor in series
- Secure jumper wire connections

b) **Check pigpiod:**

```
systemctl status pigpiod
sudo systemctl restart pigpiod
```

c) **Test individual GPIO:**

```bash
# Export GPIO 22 (red LED)
echo 22 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio22/direction

# Turn on
echo 1 > /sys/class/gpio/gpio22/value

# Turn off
echo 0 > /sys/class/gpio/gpio22/value

# Cleanup
echo 22 > /sys/class/gpio/unexport
```

d) **LED or resistor failure:**

- Test LED with multimeter
- Replace LED if needed
- Check resistor value (should be 220Ω)

### 4. Servo Not Moving

**Symptoms:**

- Flag doesn't raise/lower
- Servo makes clicking noise
- Servo jitters

**Diagnosis:**

```
# Run hardware test
python3 hardware_test.py

# Check pigpiod
systemctl status pigpiod
```

**Solutions:**

a) **Check power supply:**

- Servo draws significant current
- Use 2.5A power supply minimum
- Check 5V pin voltage with multimeter

b) **Check servo connection:**

- Red wire to 5V
- Brown/Black wire to GND
- Orange/Yellow wire to GPIO 16

c) **Adjust pulse width:** Edit `iss_tracker.py`:

```
SERVO_UP = 530    # Try values 500-700
SERVO_DOWN = 1530 # Try values 1400-1600
```

d) **Test servo independently:**

```
import pigpio
import time

pi = pigpio.pi()
pi.set_servo_pulsewidth(16, 1000)  # Middle position
time.sleep(2)
pi.set_servo_pulsewidth(16, 500)   # One end
time.sleep(2)
pi.set_servo_pulsewidth(16, 1500)  # Other end
pi.stop()
```

**5. WiFi Portal Not Accessible**

**Symptoms:**

- Can connect to PieSS-Setup but can't access portal
- Browser shows "connection refused"
- Portal page doesn't load

**Diagnosis:**

```
# Check AP services
systemctl status hostapd
systemctl status dnsmasq
systemctl status wifi_portal

# Check if portal is listening
sudo netstat -tulpn | grep 8080

# Check wlan0 configuration
ip addr show wlan0
```

**Solutions:**

a) **Services not running:**

```
sudo systemctl start hostapd
sudo systemctl start dnsmasq
sudo systemctl start wifi_portal
```

b) **Wrong IP address:**

```
# Set correct IP
sudo ip addr flush dev wlan0
sudo ip addr add 192.168.4.1/24 dev wlan0
sudo ip link set wlan0 up
```

c) **Port conflict:**

```
# Check what's using port 8080
sudo lsof -i :8080

# Kill conflicting process
sudo kill <PID>
```

d) **Firewall blocking:**

```
# If UFW is enabled
sudo ufw allow 8080/tcp
```

### 6. ISS Tracker Not Alerting

**Symptoms:**

- Tracker running but no alerts
- No passes found

**Diagnosis:**

```
# Check pass calculations
cd ~/PieSS/2025/v2
source venv/bin/activate
python3 check_passes.py

# Check logs for pass info
sudo journalctl -u iss_tracker | grep "Next visible pass"
```

**Solutions:**

a) **No visible passes in next 24 hours:**

- ISS passes vary by location and time
- Check again tomorrow
- Verify passes with ISS Detector app

b) **Location detection failed:**

```
# Check logs
sudo journalctl -u iss_tracker | grep "location"

# Manually test geolocation
python3 -c "import requests; print(requests.get('https://ipapi.co/json/').json())"
```

c) **Time/timezone issue:**

```
# Check system time
timedatectl

# Set correct timezone
sudo timedatectl set-timezone America/Toronto
```

d) **TLE data stale:**

```
# Remove cached TLE data
cd ~/PieSS/2025/v2
rm stations.tle

# Restart tracker to re-download
sudo systemctl restart iss_tracker
```

## Log Files

**Location:** Logs are stored in systemd journal

**Viewing Logs:**

```
# ISS Tracker logs
sudo journalctl -u iss_tracker -f

# WiFi Portal logs
sudo journalctl -u wifi_portal -f
cat /var/log/piess-portal.log

# Boot Decider logs
sudo journalctl -u boot_decider -f

# All PieSS logs
sudo journalctl -t boot_decider -t iss_tracker -t wifi_portal -f

# View logs since last boot
sudo journalctl -u iss_tracker -b

# View only errors
sudo journalctl -u iss_tracker -p err -n 50
```

```
# Export logs to file
sudo journalctl -u iss_tracker > ~/iss_logs.txt
```

## Hardware Diagnostics

### LED Test Sequence

```
cd ~/PieSS/2025/v2
source venv/bin/activate
python3 hardware_test.py
```

Expected output:

```
Starting hardware test...
Testing North LED (GPIO 5)...
Testing East LED (GPIO 6)...
[etc...]
Hardware test complete!
```

### Manual GPIO Testing

Test individual components:

```python
import pigpio
import time

pi = pigpio.pi()

# Test LED (GPIO 22)
pi.write(22, 1)  # On
time.sleep(1)
pi.write(22, 0)  # Off

# Test Servo
pi.set_servo_pulsewidth(16, 1000)
time.sleep(2)
pi.set_servo_pulsewidth(16, 0)  # Stop signal

pi.stop()
```

## Network Diagnostics

### Check Connectivity

```
# Test internet
ping -c 3 8.8.8.8

# Test DNS
nslookup google.com

# Show routing table
ip route show

# Show WiFi connections
nmcli con show
```

#### Check AP Mode

```
# Check hostapd
systemctl status hostapd

# Check dnsmasq
systemctl status dnsmasq

# Check wlan0
ip addr show wlan0

# Check DHCP leases
cat /var/lib/misc/dnsmasq.leases
```

### Performance Monitoring

```
# CPU usage
top -u piess

# Memory usage
free -h

# Disk space
df -h

# Temperature
vcgencmd measure_temp

# System load
uptime
```

---

## Maintenance

### Regular Maintenance Tasks

#### Daily (Automated)

- TLE data refresh (every 12 hours)
- Pass calculations
- Hardware state management

#### Weekly (Recommended)

- Check system logs for errors
- Verify disk space
- Test hardware components

#### Monthly (Recommended)

- Update system packages
- Review and clean logs
- Backup configuration files

### System Updates

```
# Update package lists
sudo apt update

# Upgrade installed packages
sudo apt upgrade -y

# Update Python packages (in venv)
cd ~/PieSS/2025/v2
source venv/bin/activate
pip list --outdated
pip install --upgrade <package_name>

# Reboot if kernel updated
sudo reboot
```

## Backup Procedures

### Configuration Backup

```
# Create backup directory
mkdir -p ~/piess_backup

# Backup service files
sudo cp /etc/systemd/system/boot_decider.service ~/piess_backup/
sudo cp /etc/systemd/system/iss_tracker.service ~/piess_backup/
sudo cp /etc/systemd/system/wifi_portal.service ~/piess_backup/
sudo cp /etc/systemd/system/pigpiod.service ~/piess_backup/

# Backup configuration files
sudo cp /etc/hostapd/hostapd.conf ~/piess_backup/
sudo cp /etc/dnsmasq.conf ~/piess_backup/
sudo cp /etc/sudoers.d/piess-sudoers ~/piess_backup/

# Backup application files
cp ~/PieSS/2025/v2/*.py ~/piess_backup/
cp ~/PieSS/2025/v2/*.sh ~/piess_backup/

# Create archive
cd ~
tar -czf piess_backup_$(date +%Y%m%d).tar.gz piess_backup/
```

### SD Card Image Backup

**Windows (Win32DiskImager):**

1. Download Win32DiskImager
2. Insert SD card in card reader
3. Select drive letter
4. Choose output file location
5. Click "Read" to create image

**macOS/Linux:**

```
# Find SD card device
lsblk  # or diskutil list on macOS

# Create image (replace /dev/sdX with your device)
sudo dd if=/dev/sdX of=~/piess_backup.img bs=4M status=progress
```

```
# Compress image
gzip ~/piess_backup.img
```

## Log Management

### View Log Sizes

```
# Journal logs
sudo journalctl --disk-usage

# Portal logs
ls -lh /var/log/piess-portal.log
```

### Clean Old Logs

```
# Keep only 1 week of journal logs
sudo journalctl --vacuum-time=1w

# Keep only 500MB of journal logs
sudo journalctl --vacuum-size=500M

# Rotate portal log
sudo mv /var/log/piess-portal.log /var/log/piess-portal.log.old
sudo systemctl restart wifi_portal
```

## Performance Optimization

### Reduce Boot Time

```
# Check boot time
systemd-analyze

# Check service startup times
systemd-analyze blame

# Disable unnecessary services
sudo systemctl disable bluetooth.service
sudo systemctl disable hciuart.service
```

### Reduce Memory Usage

```
# Check memory
free -h

# Reduce GPU memory (edit /boot/config.txt)
gpu_mem=16  # Minimal, since no display needed

# Disable swap if not needed
sudo dphys-swapfile swapoff
sudo dphys-swapfile uninstall
sudo systemctl disable dphys-swapfile
```

## Hardware Maintenance

### LED Replacement

If an LED fails:

1. Power off Raspberry Pi
2. Remove faulty LED
3. Check LED polarity (long leg = +)
4. Insert new LED
5. Verify resistor still present (220Ω)
6. Power on and test with `hardware_test.py`

### Servo Maintenance

Servos can wear out over time:

- Listen for grinding noises
- Check for smooth movement
- Replace if erratic or weak
- Use same voltage rating (5V)

### Connection Maintenance

Periodically check:

- Jumper wire connections (can loosen)
- Breadboard connections
- Solder joints (if using perfboard)
- Power supply cable

## Factory Reset

To completely reset PieSS:

```
# Stop all services
sudo systemctl stop iss_tracker
sudo systemctl stop wifi_portal
sudo systemctl stop boot_decider

# Disable services
sudo systemctl disable iss_tracker
sudo systemctl disable wifi_portal
sudo systemctl disable boot_decider

# Remove service files
sudo rm /etc/systemd/system/boot_decider.service
sudo rm /etc/systemd/system/iss_tracker.service
sudo rm /etc/systemd/system/wifi_portal.service

# Remove configuration files
sudo rm /etc/hostapd/hostapd.conf
sudo rm /etc/dnsmasq.conf
sudo rm /etc/sudoers.d/piess-sudoers

# Remove application files
rm -rf ~/PieSS

# Clean NetworkManager connections
nmcli con delete "PieSS-Setup"
```

```
# Reboot
sudo reboot
```

Then reinstall from scratch using the installation instructions.

---

## Technical Reference

### Satellite Tracking Mathematics

#### Two-Line Element (TLE) Format

TLE files contain orbital parameters:

```
ISS (ZARYA)
1 25544U 98067A   25005.12345678  .00002182  00000-0  48726-4 0  9990
2 25544  51.6416 123.4567 0003456  34.5678 325.5678 15.51234567890123
```

**Line 1:**

- Satellite number (25544)
- Epoch (year and day)
- Drag coefficient
- Ephemeris type

**Line 2:**

- Inclination (51.6416°)
- Right ascension (123.4567°)
- Eccentricity (0.0003456)
- Argument of perigee (34.5678°)
- Mean anomaly (325.5678°)
- Mean motion (15.51 orbits/day)

#### Pass Prediction Algorithm

1. **Load TLE data** for ISS
2. **Set observer location** (lat, lon, elevation)
3. **Generate time range** (typically 24 hours)
4. **Calculate position** for each time step
5. **Find local maxima** in elevation (pass peaks)
6. **Filter by minimum elevation** (15 degrees)
7. **Check night-time condition** (between sunset and sunrise)
8. **Calculate azimuth** for directional LEDs

#### Coordinate Systems

**Horizontal Coordinates (Alt-Az):**

- Altitude (elevation): 0° to 90° above horizon
- Azimuth: 0° to 360° clockwise from North

**Geographic Coordinates:**

- Latitude: -90° (South) to +90° (North)

- Longitude: -180° (West) to +180° (East)
- Elevation: meters above sea level

## GPIO Technical Details

#### Electrical Specifications

**Raspberry Pi 3B GPIO:**

- Logic levels: 3.3V (HIGH), 0V (LOW)
- Maximum current per pin: 16mA
- Total current all GPIO: 50mA maximum
- Input impedance: ~50 kΩ pull-up/down

**LED Current Calculations:**

For RED LED on GPIO 22:

```
V_gpio = 3.3V
V_led = 2.0V (red LED forward voltage)
R = 220 Ohm
I = (V_gpio - V_led) / R
I = (3.3 - 2.0) / 220 = 5.9 mA
```

**Why 220 Ohm Resistors:**

- Provides safe current (5-6 mA)
- Well below GPIO maximum (16mA)
- Provides good LED brightness
- Standard value, easily available
- Calculated value: ~215 Ohm
- Standard value used: 220Ω (more conservative, ~6mA)

**LED Specifications:**

| Color | Vf (typ) | Current | Resistor |
|---|---|---|---|
| Red | 2.0V | 15mA | 220 Ohm |
| Yellow | 2.1V | 15mA | 220 Ohm |
| Green | 3.0V | 15mA | 220 Ohm |

Using 220 Ohm for all LEDs:

- Red: (3.3-2.0)/220 = 5.9mA (good brightness)
- Yellow: (3.3-2.1)/220 = 5.5mA (good brightness)
- Green: (3.3-3.0)/220 = 1.4mA (dimmer, but safe and visible)

## 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

```
# 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

```
# 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

```
# 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

```
# 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 2024)

- Complete rewrite with portable WiFi configuration
- Progressive LED countdown with acceleration
- Sunrise/sunset based night detection
- Hardware self-test on startup
- LED status indicators for AP mode and scanning
- Automated installation script
- 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/CPowerMav/PieSS
- Issues: https://github.com/CPowerMav/PieSS/issues

**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 the PieSS implementation located in `/home/piess/PieSS/2025/v2/`. For updates and latest information, refer to the GitHub repository at https://github.com/CPowerMav/PieSS*