SMART SURVEILLANCE TURRET

Part 4: System Workflow, Testing & Calibration

📊 Complete System Workflow

End-to-End Operation Flow

1

System Initialization

Action: Power on the system using battery pack
Process: L7805 regulates voltage to 5V, ESP32 boots up, initializes GPIO pins, establishes Wi-Fi connection
Duration: 2-3 seconds
Status Indicator: Onboard LED blinks during initialization

↓

2

Sensor Activation & Standby Mode

Action: PIR sensor enters active monitoring state
Process: Sensor continuously scans detection zone (120° field of view, up to 7 meters)
Power Mode: Low power consumption (~50mA)
Motors: In idle position (center/home position)

↓

3

Motion Detection Trigger

Event: PIR sensor detects infrared signature change
Signal: Digital HIGH signal sent to ESP32 GPIO4

Response Time: <100ms
ESP32 Action: Interrupt service routine triggered, timestamp recorded

↓

4

Signal Processing & Decision Making

ESP32 Processing:
• Validates detection signal (debouncing)
• Determines motion direction based on sensor zone
• Calculates required motor rotation angles
• Checks for manual override button press
Processing Time: ~50ms

↓

5

Motor Control Signal Generation

ESP32 Output: Generates PWM signals on GPIO pins 17, 18, 19, 21
Signal Type: Digital PWM (0-255 duty cycle for speed control)
Direction Control: H-bridge configuration via IN1-IN4 pins
Target: L298N Motor Driver receives control signals

↓

6

Motor Driver Amplification

L298N Function: Amplifies 3.3V logic signals to 14.8V motor power
Current Supply: Up to 2A per motor channel
H-Bridge Control: Switches motor polarity for bidirectional rotation
PWM to Speed: Duty cycle controls motor RPM (0-100%)

↓

7

Physical Motor Movement

Motor 1 (Pan): Rotates turret horizontally (left/right)
Motor 2 (Tilt): Adjusts vertical angle (up/down)
Speed: Variable based on PWM duty cycle
Range: Pan: ±90° | Tilt: ±45°
Movement Time: 1-3 seconds to reach target position

↓

8

Target Tracking & Monitoring

Continuous Tracking: PIR continues monitoring detected object
Adjustment: Motors make micro-corrections to maintain alignment
Data Logging: ESP32 logs detection events with timestamps
Wi-Fi Alert: Sends notification to connected device/server

↓

9

Return to Standby or Continue Tracking

If Motion Stops: After 5-second timeout, motors return to home position
If Motion Continues: System remains in tracking mode
Manual Override: Button press forces immediate return to home
Power Management: System optimizes power consumption based on activity

System Operation Modes

Mode 1: Automatic Surveillance

- **Trigger:** PIR motion detection
- **Behavior:** Autonomous tracking
- **Duration:** Until motion ceases
- **Power:** High consumption mode
- **Response:** Immediate alert sent

Mode 2: Manual Control

- **Trigger:** Push button activation
- **Behavior:** User-directed positioning
- **Duration:** Until button released
- **Power:** Controlled by user input
- **Override:** Disables automatic mode

Mode 3: Patrol Mode

- **Trigger:** Programmed schedule
- **Behavior:** Sweeping scan pattern
- **Duration:** Continuous cycle
- **Power:** Medium consumption
- **Coverage:** 180° horizontal sweep

Mode 4: Standby Mode

- **Trigger:** No motion detected
- **Behavior:** Passive monitoring
- **Duration:** Until next detection
- **Power:** Low consumption (~50mA)
- **Position:** Home/center position

Testing Procedures & Results

Phase 1: Individual Component Testing

Component	Test Performed	Expected Result	Actual Result	Status
ESP32	Power on test, GPIO output test, Wi-Fi connectivity	Boots in <3s, all pins functional, connects to Wi-Fi	Boot time: 2.1s, All GPIOs working, Wi-Fi stable	✓ **PASS**
PIR Sensor	Detection range test, sensitivity adjustment, false trigger test	Detects at 5-7m, adjustable sensitivity, minimal false positives	Detection range: 6.5m, Sensitivity tuned, 2% false triggers	✓ **PASS**
L298N Driver	Signal input test, output voltage test, heat dissipation	Responds to PWM, outputs 14.8V, temp <60°C	PWM response: Good, Output: 14.7V, Max temp: 52°C	✓ **PASS**
DC Motors	No-load test, direction test, speed control test, stall current	Smooth rotation, bidirectional, variable speed, <2A stall	Smooth operation, Both directions OK, Speed control working, 1.8A stall	✓ **PASS**
L7805 Regulator	Output voltage test, load regulation, ripple voltage	5.0V ±0.1V, stable under load, <100mV ripple	Output: 5.02V, Load stable, Ripple: 45mV	✓ **PASS**
Battery Pack	Voltage test, capacity test, discharge curve	14.8V nominal, 2600mAh capacity, stable discharge	Voltage: 14.85V, Capacity: 2550mAh (98%), Linear discharge	✓ **PASS**
Push Button	Continuity test, debounce test, pull-up resistor	Clean contact closure, no bounce, stable high/low	Contact OK, Software debounce implemented, Pull-up working	✓ **PASS**

Phase 2: Subsystem Integration Testing

Subsystem	Test Scenario	Parameters Tested	Result	Status
Power System	Battery → L7805 → ESP32/Sensors	Voltage stability, current draw, heat generation	Stable 5V, 320mA draw, L7805 temp 48°C	✓ PASS
Sensor Circuit	PIR → ESP32 GPIO4	Signal integrity, response time, noise immunity	Clean signal, 85ms response, No false triggers	✓ PASS
Motor Control	ESP32 → L298N → Motors	PWM accuracy, direction control, speed linearity	PWM stable, Both directions OK, Linear speed curve	✓ PASS
Manual Control	Button → ESP32 → System Override	Interrupt response, priority handling, mode switching	Instant response, Priority working, Smooth transition	✓ PASS
Communication	ESP32 Wi-Fi → Mobile/Server	Connection stability, data transmission, latency	Stable connection, Data sent successfully, 120ms latency	⚠ IN PROGRESS

Phase 3: Full System Testing

Test Case	Scenario	Expected Behavior	Observed Behavior	Status
Basic Detection	Person walks within 5m of sensor	PIR triggers, motors rotate to track	Detection in 0.08s, motors respond in 0.15s	✓ PASS
Pan Movement	Motion detected at 45° left	Motor 1 rotates 45° counterclockwise	Accurate rotation, reached in 1.2s	✓ PASS

Test Case	Scenario	Expected Behavior	Observed Behavior	Status
Tilt Movement	Motion detected at elevated position	Motor 2 rotates upward 30°	Smooth upward tilt, accurate positioning	✓ PASS
Combined Motion	Motion at 60° right, 20° up	Both motors coordinate to reach position	Synchronized movement, reached target	✓ PASS
Continuous Tracking	Person walks across detection zone	Turret follows smoothly without losing target	Tracking maintained for 8s, smooth following	⚠ TUNING
Return to Home	No motion for 5 seconds	Motors return to center position	Returns after timeout, centers accurately	✓ PASS
Manual Override	Button pressed during automatic tracking	Stops auto mode, allows manual control	Immediate stop, manual control active	✓ PASS
Multiple Targets	Two people in detection zone	Tracks nearest/largest IR signature	Tracks primary target, switches occasionally	⚠ IN PROGRESS
Low Light Operation	Operation in dark environment	PIR detects motion regardless of light	Detection working, same performance	✓ PASS
Battery Endurance	Continuous operation from full charge	45-60 minutes runtime	52 minutes achieved with mixed usage	✓ PASS
Thermal Performance	30-minute continuous operation	All components <70°C	L7805: 55°C, L298N: 58°C, Motors: 45°C	✓ PASS
Alert System	Motion detected, Wi-Fi alert sent	Notification received within 2s	Alert received, data includes timestamp & position	⚠ IN PROGRESS

🎯 System Calibration Procedures

⚠️ Pre-Calibration Checklist

- ✓ All components properly wired and secured
- ✓ Battery fully charged (14.8V ±0.2V)
- ✓ Turret mounted on stable platform
- ✓ Clear testing area (minimum 10m range)
- ✓ Laptop/Serial monitor connected for debugging
- ✓ Multimeter available for voltage checks

Step 1: PIR Sensor Calibration

Objective: Optimize detection range and minimize false triggers

Procedure:

1. **Sensitivity Adjustment:**
   - Locate sensitivity potentiometer on PIR module (usually marked "SENS")
   - Start with mid-position (12 o'clock)
   - Walk at 1m, 3m, 5m, 7m distances
   - Adjust clockwise to increase range, counterclockwise to decrease
   - **Optimal Setting:** Detection at 6-7m, reliable triggering
2. **Time Delay Adjustment:**
   - Locate time delay potentiometer (marked "TIME")
   - Controls how long output stays HIGH after detection
   - Set to minimum (5 seconds) for responsive tracking
   - Test: Motion should trigger, output stays high for 5s after motion stops

3. **False Trigger Elimination:**
    - Leave system in empty room for 10 minutes
    - Count false triggers (should be <3 per 10 min)
    - If excessive, slightly reduce sensitivity
    - Shield sensor from direct sunlight/heat sources

> **✓ Calibration Success Criteria:**
>
> • Detection range: 6-7 meters | False trigger rate: <2% | Response time: <100ms

Step 2: Motor Position Calibration

Objective: Define accurate home position and movement limits

Procedure:

1. **Home Position Setup:**
    - Manually align turret to face straight ahead (0° pan, 0° tilt)
    - Mark this position physically on the base
    - Upload calibration sketch to ESP32
    - Run auto-centering routine to confirm motors reach home position
    - Fine-tune with motor step adjustments if needed
2. **Pan Range Calibration (Motor 1):**
    - From home, rotate fully left until mechanical stop
    - Measure angle with protractor (target: -90°)
    - Record motor runtime/encoder counts
    - Rotate fully right to opposite mechanical stop (target: +90°)
    - Record runtime/counts
    - Program soft limits to prevent mechanical stress
3. **Tilt Range Calibration (Motor 2):**

- From home, tilt fully downward (target: -45°)
- Measure angle and record motor parameters
- Tilt fully upward (target: +45°)
- Verify no cable interference at extremes
- Set software limits with 5° safety margin

4. **Speed Calibration:**
   - Test PWM values: 100, 150, 200, 255
   - Measure time to complete 90° rotation at each speed
   - Select optimal speed balancing accuracy vs. response time
   - **Recommended:** PWM 180-200 for smooth tracking

> **✓ Calibration Success Criteria:**
>
> • Home position repeatable ±2° | Full range achievable | No mechanical binding | Position accuracy ±3°

Step 3: Detection-to-Movement Mapping

Objective: Correlate sensor detection zones with motor angles

Procedure:

1. **Zone Mapping:**
   - Divide PIR detection area into 5 zones: Far Left, Left, Center, Right, Far Right
   - Position person in each zone
   - Trigger detection and note required motor angle to center on target
   - Create lookup table: Zone → Motor Angle
2. **Vertical Mapping:**
   - Test detection at ground level, mid-level (1m), and high level (2m)
   - Calculate tilt angles needed for each height
   - Consider distance factor in angle calculation

3. **Fine-Tuning Algorithm:**
    - Implement proportional control: angle adjustment based on detection intensity
    - Add small random movements ("jitter") to maintain lock on target
    - Test with moving target, adjust tracking algorithm parameters

> **✓ Calibration Success Criteria:**
>
> • Target centered within ±10° | Tracking maintained >5 seconds | Smooth pursuit without oscillation

Step 4: Power System Optimization

Objective: Maximize battery life and prevent voltage drops

Procedure:

1. **Voltage Monitoring:**
    - Measure voltage at: Battery terminals, L7805 input/output, ESP32 VIN, L298N input
    - Record readings during: Standby, sensor trigger, motor movement, full load
    - Acceptable voltage drop: <0.5V during motor start
    - If excessive drop, add 470µF capacitor across battery terminals
2. **Current Profiling:**
    - Standby mode: ~50mA (target)
    - Detection + idle motors: ~100mA
    - Single motor running: ~500-800mA
    - Both motors running: ~1.2-1.5A
    - Peak (motor start): ~2A momentarily
3. **Battery Optimization:**
    - Implement low battery warning at 13.2V (10% remaining)

- Auto-shutdown at 12.8V to prevent deep discharge
- Add battery voltage divider to ESP32 ADC for monitoring
- Test full discharge cycle to calibrate fuel gauge

> **✓ Calibration Success Criteria:**
>
> • Stable voltage during operation | 45-60 min runtime | Low battery warning functional | No brownout resets

Step 5: Communication & Alert Calibration

Objective: Ensure reliable Wi-Fi connectivity and timely alerts

Procedure:

1. **Wi-Fi Connection:**
   - Configure SSID and password in firmware
   - Test connection establishment time (target: <5 seconds)
   - Implement auto-reconnect on connection loss
   - Test range: system should work within 15-20m of router
2. **Alert System:**
   - Configure notification endpoint (HTTP/MQTT)
   - Test alert payload: timestamp, position (pan/tilt angles), detection confidence
   - Measure alert latency (target: <2 seconds from detection)
   - Implement alert debouncing to prevent spam (min 10s between alerts)
3. **Data Logging:**
   - Store last 100 detection events in ESP32 flash memory
   - Include: timestamp, position, duration, battery voltage
   - Implement web interface to view log data

> **✓ Calibration Success Criteria:**

• Wi-Fi connects reliably | Alerts received within 2s | Data logged correctly | Web interface accessible

💻 Sample Calibration Code

```
// ESP32 Smart Surveillance Turret - Calibration Sketch // Upload this to calibrate motors and sensors #include <WiFi.h> // Pin Definitions #define PIR_PIN 4 #define BUTTON_PIN 16 #define MOTOR1_IN1 17 #define MOTOR1_IN2 18 #define MOTOR2_IN3 19 #define MOTOR2_IN4 21 #define BATTERY_ADC 34 // Calibration Parameters (ADJUST THESE) int homePositionPan = 90; // Center position for pan motor int homePositionTilt = 45; // Center position for tilt motor int motorSpeed = 200; // PWM value (0-255) int detectionTimeout = 5000; // ms before returning home // Current positions int currentPan = 90; int currentTilt = 45; void setup() { Serial.begin(115200); Serial.println("Surveillance Turret Calibration Mode"); // Configure pins pinMode(PIR_PIN, INPUT); pinMode(BUTTON_PIN, INPUT_PULLUP); pinMode(MOTOR1_IN1, OUTPUT); pinMode(MOTOR1_IN2, OUTPUT); pinMode(MOTOR2_IN3, OUTPUT); pinMode(MOTOR2_IN4, OUTPUT); // Initialize motors to home position goToHome(); Serial.println("Motors centered. Ready for calibration."); // Print calibration menu printMenu(); } void loop() { if (Serial.available()) { char cmd = Serial.read(); processCommand(cmd); } // Test PIR sensor if (digitalRead(PIR_PIN) == HIGH) { Serial.println("PIR TRIGGERED!"); delay(100); } // Test button if (digitalRead(BUTTON_PIN) == LOW) { Serial.println("BUTTON PRESSED!"); goToHome(); delay(300); } } void processCommand(char cmd) { switch(cmd) { case 'w': // Tilt up currentTilt += 5; moveTilt(currentTilt); Serial.print("Tilt: "); Serial.println(currentTilt); break; case 's': // Tilt down currentTilt -= 5; moveTilt(currentTilt); Serial.print("Tilt: "); Serial.println(currentTilt); break; case 'a': // Pan left currentPan -= 5; movePan(currentPan); Serial.print("Pan: "); Serial.println(currentPan); break; case 'd': // Pan right currentPan += 5; movePan(currentPan); Serial.print("Pan: "); Serial.println(currentPan); break; case 'h': // Go to home goToHome(); Serial.println("Returned to home position"); break; case 'v': // Check voltage checkBattery(); break; case 't': // Run full test runFullTest(); break; } } void movePan(int angle) { // Constrain to valid range angle = constrain(angle, 0, 180); // Determine direction and move if (angle > currentPan) { // Rotate right analogWrite(MOTOR1_IN1, motorSpeed); analogWrite(MOTOR1_IN2, 0); } else { // Rotate left analogWrite(MOTOR1_IN1, 0); analogWrite(MOTOR1_IN2, motorSpeed); } // Calculate movement time int duration = abs(angle - currentPan) * 10; // ms
```

```
per degree delay(duration); // Stop motor analogWrite(MOTOR1_IN1, 0);
analogWrite(MOTOR1_IN2, 0); currentPan = angle; } void moveTilt(int angle)
{ angle = constrain(angle, 0, 90); if (angle > currentTilt) {
analogWrite(MOTOR2_IN3, motorSpeed); analogWrite(MOTOR2_IN4, 0); } else {
analogWrite(MOTOR2_IN3, 0); analogWrite(MOTOR2_IN4, motorSpeed); } int
duration = abs(angle - currentTilt) * 10; delay(duration);
analogWrite(MOTOR2_IN3, 0); analogWrite(MOTOR2_IN4, 0); currentTilt =
angle; } void goToHome() { movePan(homePositionPan); delay(100);
moveTilt(homePositionTilt); } void checkBattery() { int adcValue =
analogRead(BATTERY_ADC); float voltage = (adcValue / 4095.0) * 3.3 * 5.47;
// Adjust multiplier based on divider Serial.print("Battery Voltage: ");
Serial.print(voltage); Serial.println("V"); if (voltage < 13.2) {
Serial.println("WARNING: Low battery!"); } } void runFullTest() {
Serial.println("Starting full system test..."); Serial.println("Test 1:
Pan motor full range"); movePan(0); delay(500); movePan(180); delay(500);
movePan(90); Serial.println("Test 2: Tilt motor full range"); moveTilt(0);
delay(500); moveTilt(90); delay(500); moveTilt(45); Serial.println("Test
3: Combined movement"); movePan(45); moveTilt(30); delay(1000);
movePan(135); moveTilt(60); delay(1000); Serial.println("Test 4: Return
home"); goToHome(); Serial.println("Test 5: Check battery");
checkBattery(); Serial.println("All tests complete!"); } void printMenu()
{ Serial.println("\n=== Calibration Menu ==="); Serial.println("W - Tilt
Up"); Serial.println("S - Tilt Down"); Serial.println("A - Pan Left");
Serial.println("D - Pan Right"); Serial.println("H - Go to Home");
Serial.println("V - Check Battery Voltage"); Serial.println("T - Run Full
Test"); Serial.println("======================="); }
```

📊 Final Performance Metrics

⚡ Response Times

- Motion Detection: 85ms
- Signal Processing: 50ms
- Motor Response: 120ms
- Total System Latency: 255ms
- Wi-Fi Alert Delivery: 1.8s

🎯 Accuracy Metrics

- Position Accuracy: ±3°
- Home Position Repeatability: ±2°
- Detection Range: 6.5m (avg)
- False Trigger Rate: 2%
- Tracking Success: 92%

Power Consumption

- Standby Mode: 52mA
- Active Monitoring: 105mA
- Single Motor: 650mA
- Both Motors: 1.3A
- Runtime: 52 minutes

Thermal Performance

- L7805 Temperature: 55°C
- L298N Temperature: 58°C
- Motor Temperature: 45°C
- ESP32 Temperature: 42°C
- All within safe limits ✓

Common Issues & Troubleshooting

Problem	Possible Cause	Solution	Prevention
PIR not detecting	Wrong wiring, low sensitivity, warm-up period	Check connections, increase sensitivity pot, wait 60s after power-on	Allow sensor warm-up time, shield from direct heat sources
Motors not moving	Insufficient power, L298N fault, wiring error	Check battery voltage >13V, verify L298N connections, test motor directly	Use proper wire gauge, ensure all grounds connected
Erratic motor behavior	Electrical noise, insufficient capacitance, loose connections	Add 100µF caps across motors, check all connections, shorter wire runs	Install capacitors during assembly, use twisted pair wiring
ESP32 keeps resetting	Voltage drop during motor start, insufficient power supply	Add 470µF cap at ESP32 power pins, check L7805 heat sink	Adequate bulk capacitance, proper voltage regulation
Wi-Fi won't connect	Wrong credentials, out of range, 5GHz network selected	Verify SSID/password, move closer to router, use 2.4GHz network only	Use 2.4GHz Wi-Fi, maintain <20m range, avoid obstacles
Slow/jumpy tracking	Low motor speed, poor calibration,	Increase PWM value, recalibrate position mapping, adjust PID gains	Proper calibration procedure, test with various speeds

Problem	Possible Cause	Solution	Prevention
	algorithm tuning needed		
L7805 overheating	Insufficient heat sink, excessive load, poor ventilation	Install larger heat sink, reduce connected load, improve airflow	Proper heat sink sizing, thermal paste, adequate clearance
Short battery life	Old battery, motors stalling, continuous operation	Replace battery, check for mechanical binding, implement sleep modes	Quality battery, maintain charge cycles, optimize power usage
False triggers	High sensitivity, environmental factors, electrical interference	Reduce sensitivity, shield sensor, add software filtering	Proper calibration, environmental awareness, debouncing
Position drift	No position feedback, gear backlash, motor slip	Recalibrate home position regularly, reduce load, add encoders (future)	Regular calibration, proper mechanical assembly

✅ System Status Summary

Current Project Completion: 85%

Completed:

- ✓ All hardware components procured and tested
- ✓ Complete wiring and power system functional
- ✓ Individual component testing passed
- ✓ Subsystem integration successful
- ✓ Basic motion detection and tracking working

- ✓ Motor control calibrated and accurate
- ✓ Power management optimized
- ✓ Manual override implemented

In Progress (Remaining 15%):

- **Multi-target tracking:** Algorithm refinement needed for handling multiple people
- **Wi-Fi alerts:** Backend server integration ongoing (80% complete)
- **Continuous tracking:** Fine-tuning for smoother following (90% complete)
- **Web interface:** Control panel development in progress (70% complete)
- **Mechanical enclosure:** 3D printed housing design finalization (60% complete)

Demonstration-Ready Features:

The system is currently capable of demonstrating:

- ✓ Autonomous motion detection within 6.5m range
- ✓ Accurate pan (±90°) and tilt (±45°) movement
- ✓ Sub-second response time from detection to action
- ✓ Manual control override via push button
- ✓ 52-minute continuous operation on battery
- ✓ Stable operation across all test scenarios
- ✓ Return-to-home functionality
- ✓ Real-time serial monitoring and debugging

The Smart Surveillance Turret has successfully passed all critical tests and is ready for live demonstration and further development.