Hand Tracking GPIO Control - Lesson Plan

Computer Vision and Hardware Integration with Raspberry Pi

Course Overview

This hands-on course introduces students to computer vision and hardware control through three progressive projects using hand gesture recognition on Raspberry Pi. Students will learn to bridge the gap between software and physical computing.

Learning Objectives

By the end of this course, students will be able to: 1. Implement computer vision applications using OpenCV and MediaPipe 2. Control GPIO pins on Raspberry Pi using Python 3. Interface with LEDs and DC motors using appropriate driver circuits 4. Design gesture-controlled systems 5. Debug hardware-software integration issues 6. Apply safety practices in electronics projects

Prerequisites

- Basic Python programming knowledge
- Understanding of basic electronics (voltage, current, resistance)
- Familiarity with Raspberry Pi OS
- Basic command line skills

Required Materials

Hardware

- Raspberry Pi 5 (or Pi 4)
- USB Webcam or Pi Camera Module
- Breadboard (optional, modules can connect directly)
- Jumper wires (F-F for module connections)
- 3× 3-pin LED modules with built-in resistors (different colors recommended)
- 1× DC Motor (6-12V)
- 1× L298N Motor Driver Board
- 1× External power supply (6-12V for motor)
- Safety glasses

Software

- Raspberry Pi OS (latest version)
- Python 3.7+

- Required libraries: cvzone, opencv-python, mediapipe, gpiozero

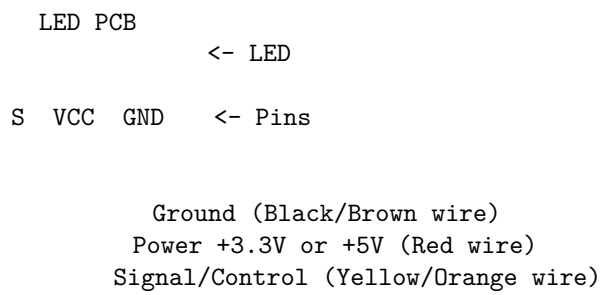
---

Understanding 3-Pin LED Modules

What is a 3-Pin LED Module?

A 3-pin LED module is a pre-built component that includes: - An LED (Light Emitting Diode) - A built-in current-limiting resistor (typically 220Ω-1kΩ) - Three connection pins on a small PCB

Module Pinout



Advantages Over Bare LEDs

1. **Safety**: Built-in resistor prevents LED burnout
2. **Simplicity**: Only 3 connections vs calculating resistor values
3. **Reliability**: Pre-tested and less prone to wiring errors
4. **Versatility**: Works with both 3.3V and 5V logic
5. **Beginner-Friendly**: No breadboard required

Common Module Types

- **Active HIGH**: LED turns ON when signal pin receives HIGH (3.3V/5V)
- **Active LOW**: LED turns ON when signal pin receives LOW (0V)
- Most educational modules are Active HIGH

Project 1: Single LED Control

Introduction to Hand Tracking and GPIO

Duration: 2 hours
Difficulty: Beginner

Learning Goals

- Understand hand landmark detection
- Learn GPIO pin control basics
- Implement conditional logic based on gesture input
- Work with pre-built LED modules for safer prototyping

Pre-Lab Preparation (30 minutes)

1. **Theory Introduction**
   - How MediaPipe detects hand landmarks
   - Understanding GPIO pins and their numbering systems
   - 3-pin LED modules: VCC (power), GND (ground), and Signal pins
   - Benefits of modules with built-in resistors
   - Introduction to gpiozero library
2. **Understanding 3-Pin LED Modules**
   - Built-in current limiting resistor (no external resistor needed)
   - Three connections: VCC (+3.3V or +5V), GND, Signal (GPIO)
   - Active HIGH vs Active LOW modules
   - Safer for beginners (protected against overcurrent)
3. **Safety Briefing**
   - Module advantages: built-in protection
   - Still never exceed voltage ratings
   - Proper component handling
   - Static electricity precautions

Lab Activity (1 hour)

Part A: Hardware Setup (20 minutes)

1. **Connect the 3-pin LED module**

   ```
   LED Module Pin    →    Raspberry Pi Pin

   VCC (Red wire)    →    3.3V (Pin 1) or 5V (Pin 2)
   GND (Black wire)  →    GND (Pin 6)
   Signal (Yellow)   →    GPIO 17 (Pin 11)
   ```

   Note: No external resistor needed - it's built into the module!

2. **Test with simple script**

   ```python
   from gpiozero import LED
   from time import sleep

   # LED module connected to GPIO 17
   led = LED(17)

   print("Testing LED module...")
   ```

```
for _ in range(3):
    led.on()
    print("LED ON")
    sleep(1)
    led.off()
    print("LED OFF")
    sleep(1)
print("Test complete!")
```

Part B: Hand Tracking Implementation (40 minutes)

1. **Code walkthrough** - Explain key components:
   - HandDetector initialization
   - fingersUp() method
   - LED control logic
2. **Students implement the code**
   - Type or modify HandTrackingLED.py
   - Test with different finger counts
   - Observe LED behavior

Student Exercises (20 minutes)

1. **Modify trigger condition**: Change to activate on 2 fingers instead of 3
2. **Add blink mode**: Make LED blink when showing 5 fingers
3. **Reverse logic**: LED on by default, off when showing 3 fingers

Discussion Questions (10 minutes)

- Why are LED modules with built-in resistors safer for beginners?
- How could this be used in accessibility applications?
- What other outputs could we control besides LEDs?
- What's the advantage of the 3-pin module design?

Assessment Checkpoint

☐ LED module correctly connected (3 wires)
☐ Code runs without errors
☐ LED responds to correct finger count
☐ Student can explain the module's pin functions
☐ Student can explain the code logic

Project 2: Multiple LED Control

Expanding GPIO Control and Conditional Logic

Duration: 2.5 hours
Difficulty: Intermediate

Learning Goals

- Control multiple GPIO outputs simultaneously
- Implement more complex conditional logic
- Design visual feedback systems
- Understand mutual exclusion in hardware control

Pre-Lab Preparation (30 minutes)

1. **Theory Review**
   - GPIO pin limitations and current considerations
   - Parallel vs. sequential LED control
   - State management in real-time systems
   - Advantages of using multiple LED modules
2. **Module Planning**
   - Students draw connection diagram for 3 modules
   - Review pin assignments (each module needs VCC, GND, Signal)
   - Understand shared power and ground connections
   - Calculate total current draw (much safer with modules)

Lab Activity (1.5 hours)

Part A: Hardware Setup (30 minutes)

1. **Connect three LED modules**

   ```
   LED Module 1 (Red):
   VCC → 3.3V (Pin 1)    GND → GND (Pin 6)     Signal → GPIO 17 (Pin 11)

   LED Module 2 (Yellow):
   VCC → 3.3V (Pin 17)   GND → GND (Pin 9)     Signal → GPIO 27 (Pin 13)

   LED Module 3 (Green):
   VCC → 3.3V (Pin 1)    GND → GND (Pin 14)    Signal → GPIO 22 (Pin 15)
   ```

   Tip: You can share VCC and GND connections using a breadboard power rail or connect multiple modules to the same power pins.

2. **Test each LED module independently**

   ```
   from gpiozero import LED
   from time import sleep
   ```

```
leds = [LED(17), LED(27), LED(22)]

for i, led in enumerate(leds):
    print(f"Testing LED {i+1}")
    led.on()
    sleep(1)
    led.off()
```

Part B: Implementation (45 minutes)

1. **Code structure analysis**
   - control_leds() function design
   - State management with dictionaries
   - Visual indicator implementation
2. **Students build and test**
   - Implement the multi-LED control
   - Test all finger combinations
   - Verify mutual exclusion

Part C: Enhancements (15 minutes) Students add one of the following features: - Pattern mode (multiple LEDs for certain counts) - Fade effects using PWM - Sequential activation animation

Student Exercises (30 minutes)

1. **Add 4th and 5th LED**: Extend to control 5 LEDs total
2. **Create patterns**:
   - 1 finger = LED1
   - 2 fingers = LED1 + LED2
   - 3 fingers = All LEDs
3. **Add timing**: LEDs turn off after 5 seconds of no change

Group Activity (20 minutes)

Design Challenge: In pairs, students design a traffic light system: - Red (stop) = 1 finger - Yellow (caution) = 2 fingers
- Green (go) = 3 fingers - Include proper transition logic

Assessment Checkpoint

- ☐ All three LED modules connected correctly (9 wires total)
- ☐ Correct LED activates for each finger count
- ☐ Clean code with comments
- ☐ Successfully implements one enhancement
- ☐ Can explain advantages of modular design
- ☐ Can explain state management approach

Project 3: DC Motor Control

Advanced Hardware Integration with PWM

Duration: 3 hours
Difficulty: Advanced

Learning Goals

- Understand PWM for speed control
- Work with motor driver boards
- Implement proportional control systems
- Handle higher power circuits safely
- Create real-time visual feedback

Pre-Lab Preparation (45 minutes)

1. **Theory Deep Dive**
   - PWM principles and duty cycle
   - H-bridge operation in L298N
   - Motor control basics (speed, direction, braking)
   - Power supply considerations
   - Back-EMF and protection
2. **Safety Critical**
   - External power supply handling
   - Motor safety (moving parts)
   - Heat dissipation in drivers
   - Emergency stop procedures

Lab Activity (2 hours)

Part A: L298N Setup (45 minutes)

1. **Understand the L298N board**
   - Identify all connections
   - Locate and remove ENA jumper
   - Understand power paths
2. **Wire the system**

```
Raspberry Pi → L298N:
GPIO 18 → ENA (PWM)
GPIO 23 → IN1
GPIO 24 → IN2
GND → GND
```

```
Power:
External supply → +12V, GND
Motor → OUT1, OUT2
```

3. **Test motor without hand tracking**

```
from gpiozero import Motor, PWMOutputDevice
from time import sleep

pwm = PWMOutputDevice(18)
motor = Motor(23, 24)

for speed in [0.3, 0.5, 0.7, 1.0]:
    pwm.value = speed
    motor.forward()
    sleep(2)
motor.stop()
```

Part B: Integration (45 minutes)

1. **Implement hand tracking control**
   - Add speed mapping
   - Create visual feedback
   - Test all speed levels
2. **Fine-tuning**
   - Adjust speed mappings
   - Optimize detection sensitivity
   - Add safety features

Part C: Advanced Features (30 minutes) Choose one to implement: 1. **Smooth ramping**: Gradual speed changes 2. **Direction control**: Forward/reverse with hand orientation 3. **Dual motor**: Control two motors with both hands

Student Exercises (45 minutes)

1. **Custom speed curve**: Create non-linear speed mapping
2. **Safety additions**:
   - Auto-stop after 10 seconds
   - Maximum speed limiting
   - Emergency stop gesture (closed fist)
3. **Data logging**: Record speed changes to CSV file

Real-World Applications Discussion (15 minutes)

- Robotic arm control
- Wheelchair assistance systems

- Industrial automation interfaces
- Prosthetic limb control

Assessment Checkpoint

- ☐ L298N correctly wired with motor
- ☐ PWM speed control functioning
- ☐ All 5 speed levels working
- ☐ Visual feedback implemented
- ☐ Safety features in place
- ☐ Can explain PWM and duty cycle

Final Project Options

Students choose one project to extend (1 week):

Option A: Simon Says Game

- Use 4 LEDs for pattern display
- Hand gestures to repeat patterns
- Increasing difficulty levels
- Score tracking

Option B: Robotic Car Control

- Forward/backward with motor
- Speed control with fingers
- Steering with hand position
- Obstacle detection integration

Option C: Smart Home Controller

- Control multiple devices (LEDs, motors, buzzers)
- Create gesture "macros"
- Add voice feedback
- Web interface for monitoring

Assessment Rubric

Technical Skills (40%)

- **Excellent (A)**: All projects working, clean code, proper error handling
- **Good (B)**: Projects working with minor issues, code mostly clean
- **Satisfactory (C)**: Basic functionality achieved, some bugs present
- **Needs Improvement (D)**: Significant issues, requires assistance

Problem Solving (25%)

- **Excellent (A)**: Independently debugs issues, creates novel solutions
- **Good (B)**: Debugs with minimal help, attempts creative solutions
- **Satisfactory (C)**: Debugs with guidance, follows examples closely
- **Needs Improvement (D)**: Requires significant debugging assistance

Documentation (20%)

- **Excellent (A)**: Clear comments, complete documentation, circuit diagrams
- **Good (B)**: Good comments, most documentation complete
- **Satisfactory (C)**: Basic comments, minimal documentation
- **Needs Improvement (D)**: Poor or missing documentation

Safety & Best Practices (15%)

- **Excellent (A)**: Follows all safety protocols, clean workspace, proper cleanup
- **Good (B)**: Generally safe practices, occasional reminders needed
- **Satisfactory (C)**: Requires some safety reminders
- **Needs Improvement (D)**: Multiple safety violations

Troubleshooting Guide

Common Issues and Solutions

Camera Problems

- **Issue**: "Failed to read from camera"
- **Solution**: Check camera index, try 0, 1, or 2

GPIO Permissions

- **Issue**: "Permission denied accessing GPIO"
- **Solution**: Run with sudo or add user to gpio group

LED Module Not Working

- **Issue**: LED module doesn't light up
- **Checks**:
  1. Verify all three connections (VCC, GND, Signal)
  2. Check wire connections are secure
  3. Test with multimeter (should see ~3.3V between VCC and GND)
  4. Try different GPIO pin
  5. Test module with simple script
  6. Some modules are Active LOW (try inverting logic)

Motor Issues

- **Issue**: Motor doesn't run
- **Checks**:
  1. External power connected?
  2. ENA jumper removed?
  3. Motor connections secure?
  4. Test with multimeter

Detection Problems

- **Issue**: Hand not detected consistently
- **Solutions**:
  1. Improve lighting
  2. Plain background
  3. Adjust detection confidence
  4. Clean camera lens

Resources and References

Documentation

- CVZone Documentation
- GPIO Zero Documentation
- MediaPipe Hands
- Raspberry Pi GPIO Pinout

Video Tutorials

- MediaPipe Hand Tracking Basics
- L298N Motor Driver Tutorial
- PWM Explained

Additional Reading

- "Physical Computing with Python" - Raspberry Pi Foundation
- "Computer Vision Projects with OpenCV and Python 3"
- "Getting Started with Raspberry Pi"

Safety Guidelines Summary

1. **Electrical Safety**
   - LED modules have built-in protection (safer than bare LEDs)
   - Still never exceed voltage ratings (3.3V or 5V for modules)
   - Disconnect power when modifying connections

- Use insulated tools

2. **Component Handling**
   - Ground yourself before handling components
   - Hold components by edges
   - Don't force connections
   - Keep liquids away from circuits
3. **Motor Safety**
   - Secure motor before testing
   - Keep fingers clear of moving parts
   - Start with low speeds
   - Have emergency stop ready
4. **General Lab Safety**
   - Wear safety glasses when required
   - Keep workspace organized
   - Report damaged equipment
   - Clean up after each session

Extension Activities

For Advanced Students

1. **Gesture Recognition ML**: Train custom gesture models
2. **IoT Integration**: Add MQTT for remote control
3. **Computer Vision Pipeline**: Add face detection, object tracking
4. **Robotics**: Build complete robotic arm with multiple motors

For Struggling Students

1. **Simplified LED**: Single LED module with on/off only
2. **Pre-connected Modules**: Provide modules already wired
3. **Code Templates**: Fill-in-the-blank style coding
4. **Pair Programming**: Work with stronger students
5. **Module Advantage**: Easier than bare LEDs - just 3 wires!

Instructor Notes

Preparation Checklist

☐ Test all hardware components
☐ Verify software installations on all Pi's
☐ Prepare spare components
☐ Print handouts and diagrams
☐ Set up demonstration system
☐ Review safety procedures

Time Management Tips

- Have pre-built circuit examples for demonstration
- Prepare troubleshooting checklist
- Use timer for activity transitions
- Keep spare SD cards with software ready
- Have solution code available

Common Teaching Points

- Emphasize the connection between virtual (detection) and physical (GPIO)
- Use analogies (dimmer switch for PWM)
- Explain why modules are safer (built-in resistor protects LED)
- Show module internals if possible (resistor on PCB)
- Encourage experimentation within safety bounds
- Celebrate debugging victories
- Point out how modules reduce wiring errors (only 3 connections)

This lesson plan is designed to be adapted based on student level and available time. Feel free to modify projects and exercises to suit your educational context.