Traffic Light Controller - Setup Guide

Shopping List

Components Needed:

- Raspberry Pi 5
- Breadboard (at least 400 tie points)
- 6x LEDs (2 red, 2 yellow, 2 green)
- 6x 330Ω resistors
- 7+ jumper wires (male-to-female for Pi connections)
- Optional: small cardboard to make "housings" for the lights

Wiring Diagram

```
Raspberry Pi 5 GPIO Layout (partial):

┌──────────┬───────┬─────────┐
│ Pin 11  │ GPIO │  17    │ ──→ Street A RED LED
│ Pin 13  │ GPIO │  27    │ ──→ Street A YELLOW LED
│ Pin 15  │ GPIO │  22    │ ──→ Street A GREEN LED
│ Pin 16  │ GPIO │  23    │ ──→ Street B RED LED
│ Pin 18  │ GPIO │  24    │ ──→ Street B YELLOW LED
│ Pin 22  │ GPIO │  25    │ ──→ Street B GREEN LED
│ Pin 6   │ GND  │        │ ──→ Common Ground (or use Pin 9, 14, 20, etc.)
└──────────┴───────┴─────────┘
```

Breadboard Layout

```
  Street A (N-S)          Street B (E-W)

  ┌─────────────┐      ┌─────────────┐
  │ ● RED     │         │ ● RED     │
  │ ● YELLOW │         │ ● YELLOW │
  │ ● GREEN  │         │ ● GREEN  │
  └─────────────┘      └─────────────┘
```

Detailed Wiring Instructions

Street A (North-South Direction):

1. **RED LED (GPIO 17)**

- Insert LED into breadboard (long leg = positive)
- Connect GPIO 17 (Pin 11) → LED long leg
- Connect LED short leg → 330Ω resistor → Ground rail

2. **YELLOW LED (GPIO 27)**
   - Insert LED into breadboard (long leg = positive)
   - Connect GPIO 27 (Pin 13) → LED long leg
   - Connect LED short leg → 330Ω resistor → Ground rail
3. **GREEN LED (GPIO 22)**
   - Insert LED into breadboard (long leg = positive)
   - Connect GPIO 22 (Pin 15) → LED long leg
   - Connect LED short leg → 330Ω resistor → Ground rail

Street B (East-West Direction):

4. **RED LED (GPIO 23)**
   - Insert LED into breadboard (long leg = positive)
   - Connect GPIO 23 (Pin 16) → LED long leg
   - Connect LED short leg → 330Ω resistor → Ground rail
5. **YELLOW LED (GPIO 24)**
   - Insert LED into breadboard (long leg = positive)
   - Connect GPIO 24 (Pin 18) → LED long leg
   - Connect LED short leg → 330Ω resistor → Ground rail
6. **GREEN LED (GPIO 25)**
   - Insert LED into breadboard (long leg = positive)
   - Connect GPIO 25 (Pin 22) → LED long leg
   - Connect LED short leg → 330Ω resistor → Ground rail
7. **Ground Connection**
   - Connect Pi Ground (Pin 6 or any GND pin) → Breadboard ground rail

Visual LED Arrangement

For best effect, arrange LEDs vertically on the breadboard:

```
Street A:         Street B:
  [RED]             [RED]
   ↓                 ↓
[YELLOW]          [YELLOW]
   ↓                 ↓
 [GREEN]           [GREEN]
```

Traffic Light Sequence

The program runs a realistic traffic light cycle:

```
Cycle Timeline (12 seconds total per full cycle):

Time  | Street A (N-S) | Street B (E-W)
------|----------------|---------------
0-5s  | GREEN          | RED
5-6s  | YELLOW         | RED
6-7s  | RED            | RED (safety buffer)
7-12s | RED            | GREEN
12-13s| RED            | YELLOW
13-14s| RED            | RED (safety buffer)
[repeat]
```

Compilation & Running

```bash
# Compile the program
gcc -o traffic_light traffic_light.c

# Run (requires root access for GPIO)
sudo ./traffic_light
```

Expected Output

```
╔════════════════════════════════════════════════════════╗
║   ARM Assembly Traffic Light Controller - Pi 5         ║
║   Two-Way Intersection Simulator                       ║
╚════════════════════════════════════════════════════════╝

Pin Configuration:
```

```
  Street A (N-S): Red=GPIO17, Yellow=GPIO27, Green=GPIO22
  Street B (E-W): Red=GPIO23, Yellow=GPIO24, Green=GPIO25

Timing: Green=5s, Yellow=1s, Safety Buffer=1s

Configuring GPIO pins...
Starting traffic light sequence... (Press Ctrl+C to exit)

[Cycle 001] Street A (N-S): GREEN  | Street B (E-W): RED
```

Troubleshooting

LEDs don't light up:

- Check LED polarity (long leg = positive)
- Verify resistor connections
- Confirm GPIO pin numbers
- Test with simple blink program first

Only some LEDs work:

- Check individual connections
- Test each LED separately with a battery
- Verify breadboard connections aren't loose

"Cannot open /dev/mem":

- Must run with `sudo`

Timing seems off:

- System load can affect timing slightly
- This is software timing (not hardware PWM)

Teaching Points

State Machine Concepts:

This traffic light is a **finite state machine** with 6 states:

1. A-Green, B-Red
2. A-Yellow, B-Red
3. A-Red, B-Red (buffer)

4. A-Red, B-Green
5. A-Red, B-Yellow
6. A-Red, B-Red (buffer)

ARM Assembly Demonstrations:

- **Bit manipulation** (BIC, ORR for GPIO control)
- **Memory-mapped I/O** (GPIO registers)
- **Batch operations** (setting/clearing multiple pins at once)

Extensions for Students:

1. **Add a pedestrian crossing button**
   - Read GPIO input
   - Interrupt current cycle
   - Show pedestrian walk signal
2. **Emergency vehicle override**
   - All lights red
   - Flash yellow for intersection
3. **Time-of-day adjustment**
   - Longer green times during rush hour
   - Flashing yellow late at night
4. **Sensor simulation**
   - Car detection (simulated with button press)
   - Adaptive timing based on traffic
5. **SOS mode**
   - Flash all lights in Morse code pattern

Safety Notes

- Always use current-limiting resistors (330Ω recommended)
- GPIO pins are 3.3V (not 5V!)
- Don't exceed 16mA per GPIO pin
- Double-check wiring before power-on

- Use proper wire gauge for connections

Fun Fact

Real traffic lights have a "all red" buffer period to prevent accidents. This is called the **intergreen period** or **clearance interval**. Our 1-second buffer simulates this safety feature!