

JarzRover iPhone-ESP32 - Component Tests

Matched to the v1.1.0 dedicated iOS ESP32 pin map. Use only with the matching [Wiring Reference](#). These programs are also supplied as separate .ino files so you do not need to copy code out of a PDF.

Work together safely

An adult checks wiring, soldering, power supplies and powered motor tests. Children can label parts, read steps and record results. Disconnect battery and USB before moving wires. Keep all wheels raised during motor tests; keep fingers, hair and loose clothing clear. Stop immediately for buzzing, a stalled motor, heat, smoke or a damaged battery. Never charge a damaged battery. Use its specified charger.

Do not apply raw battery voltage to 5V, 3V3 or signal pins. Never feed motor current through the controller board, USB cable or a breadboard. Photos show mechanical placement; board labels and the wiring table determine electrical connections.

Setup and test order

Install esp32 by Espressif Systems **2.0.17**, select **ESP32 Dev Module**, then your board USB port. These sketches use the 2.x LEDC API; BLE comes with this board package. The wiring is for classic ESP-WROOM-32, not C3/S3.

Extract the ZIP. In Arduino IDE, open the named .ino inside tests/iphone/. Keep each sketch inside its same-named folder; do not paste several tests into one sketch. Click Verify, then Upload, then open Serial Monitor at **115200 baud**. Open one complete test at a time. Disconnect motor power before uploads. Sensor tests hold the four motor-control pins LOW.

Run tests in the order below. Test code replaces normal firmware. After all tests, restore the normal program using the last section before connecting Robot.

Pin reference

Board pin	Connect to	Purpose / check
GPIO16	L298N IN1	Left motor input 1
GPIO17	L298N IN2	Left motor input 2
GPIO25	L298N IN3	Right motor input 1
GPIO26	L298N IN4	Right motor input 2
GPIO32	HC-SR04 TRIG	Verify 3.3 V trigger is recognized
GPIO33	Conditioned HC-SR04 ECHO	Divider or verified level shifter required
GPIO34	Conditioned left speed DO	Input only; no internal pull resistor
GPIO35	Conditioned right speed DO	Input only; no internal pull resistor
GND	L298N GND and sensor grounds	Common signal reference

This edition replaces the old ESP32 PDF map. Do not use motor 13/12 and 15/2, sonar 5/18 or speed 35/22 from that earlier document. These tests use motors 16/17, 25/26; sonar 32/33; speed 34/35.

Board and USB serial

File: tests/iphone/01_serial/01_serial.ino

Disconnect the motor battery. Upload and open Serial Monitor at 115200 baud.

PASS: READY repeats about once each second.

NEED HELP: If blank or garbled, select the correct USB port, data cable and 115200 baud; press Reset after opening Serial Monitor.

```
// JARzLabs iPhone-ESP32 component test, documentation v1.1.0.
// Adult supervision. Disconnect motor power before uploading.
// Matches the dedicated openbot_ios firmware pin map.
#include <Arduino.h>
const int motorPins[] = {16, 17, 25, 26};
void holdMotorsOff() {
  for (int pin : motorPins) {
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }
}
void setup() {
  holdMotorsOff();
  Serial.begin(115200);
}
void loop() {
  Serial.println("JARzLabs board test: READY (motors OFF)");
  delay(1000);
}
```

Ultrasonic sensor

File: tests/iphone/02_sonar/02_sonar.ino

Keep motor power off. Confirm TRIG and ECHO from the pin table. Verify the ECHO divider/level shifter before connecting GPIO33. Move a flat target slowly between roughly 15 and 50 cm.

PASS: Distance changes sensibly as the target moves; a timeout prints NO ECHO.

NEED HELP: Check supply, common ground, pin order and target angle. NO ECHO does not mean zero distance.

```
// JARzLabs iPhone-ESP32 component test, documentation v1.1.0.
// Adult supervision. Disconnect motor power before uploading.
// Matches the dedicated openbot_ios firmware pin map.
#include <Arduino.h>
const int motorPins[] = {16, 17, 25, 26};
void holdMotorsOff() {
  for (int pin : motorPins) {
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }
}
const int TRIG = 32, ECHO = 33;
void readSonar() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  unsigned long us = pulseIn(ECHO, HIGH, 30000UL);
  if (us == 0) Serial.println("NO ECHO - check wiring/target");
  else {
    Serial.print("Distance cm: ");
    Serial.println(us * 0.0343f / 2.0f, 1);
  }
}
void setup() {
  holdMotorsOff();
  Serial.begin(115200);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
}
void loop() {
  readSonar();
  delay(250);
}
```

Left and right wheel sensors

File: tests/iphone/03_speed/03_speed.ino

Keep motor power off. Rotate only the left wheel slowly by hand, then only the right. Pause with a slot and a solid part in the sensor to see raw 0/1.

PASS: Only the relevant edge count increases; raw level changes with disk position. Counts are transitions, not revolutions or RPM.

NEED HELP: Check DO, alignment, supply and ground. GPIO34/35 have no internal pull resistors; match external bias/level shifting to the actual module. Counts rising on a stationary wheel suggest a floating/noisy signal.

```
// JARzLabs iPhone-ESP32 component test, documentation v1.1.0.
// Adult supervision. Disconnect motor power before uploading.
// Matches the dedicated openbot_ios firmware pin map.
#include <Arduino.h>
const int motorPins[] = {16, 17, 25, 26};
void holdMotorsOff() {
  for (int pin : motorPins) {
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }
}
const int LEFT = 34, RIGHT = 35;
volatile unsigned long leftEdges = 0, rightEdges = 0;
void IRAM_ATTR leftChanged() { leftEdges++; }
void IRAM_ATTR rightChanged() { rightEdges++; }
void setupSpeed() {
  // ESP32 GPIO34/35 need external bias for open-collector outputs.
  pinMode(LEFT, INPUT);
  pinMode(RIGHT, INPUT);
  attachInterrupt(digitalPinToInterrupt(LEFT), leftChanged, CHANGE);
  attachInterrupt(digitalPinToInterrupt(RIGHT), rightChanged, CHANGE);
}
void readSpeed() {
  noInterrupts();
  unsigned long l = leftEdges, r = rightEdges;
  interrupts();
  Serial.print("L edges="); Serial.print(l);
  Serial.print(" R edges="); Serial.print(r);
  Serial.print(" | raw L="); Serial.print(digitalRead(LEFT));
  Serial.print(" R="); Serial.println(digitalRead(RIGHT));
}
void setup() {
  holdMotorsOff();
  Serial.begin(115200);
  setupSpeed();
}
void loop() {
  readSpeed();
  delay(250);
}
```

Short motor pulses - adult helper required

File: tests/iphone/04_motor/04_motor.ino

Raise every wheel and clear the area. With power off, verify wiring and driver enables. Upload; open Serial Monitor at 115200, choose No line ending, then have the adult enable motor power. Send one l, L, r or R command at a time. Keep the switch within reach.

PASS: Only the selected side moves, its two wheels agree, opposite-case commands reverse it, and it stops after at most 700 ms. Send x during a pulse to test an earlier stop.

NEED HELP: Stop for buzzing or heat. This low-duty test may not overcome static friction; do not increase power blindly. Check mechanics, power, enables and wiring. The letters select electrical direction, not calibrated forward. Never leave queued commands in Serial Monitor.

```
// JARzLabs iPhone-ESP32 component test, documentation v1.1.0.
// Adult supervision. Disconnect motor power before uploading.
// Matches the dedicated openbot_ios firmware pin map.
#include <Arduino.h>
const int motorPins[] = {16, 17, 25, 26};
void holdMotorsOff() {
  for (int pin : motorPins) {
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }
}
// Send l / L for left in opposite directions, r / R for right.
// Each command runs for at most 700 ms; x stops immediately.
unsigned long started = 0;
bool running = false;
void pwm(int channel, int duty) { ledcWrite(channel, duty); }
void stopAll() {
  for (int i = 0; i < 4; i++) pwm(i, 0);
  running = false;
}
void setup() {
  holdMotorsOff();
  Serial.begin(115200);
  for (int i = 0; i < 4; i++) {
    ledcSetup(i, 5000, 8);
    ledcAttachPin(motorPins[i], i);
    ledcWrite(i, 0);
  }
  stopAll();
  Serial.println("Raise wheels. l/L=left r/R=right x=stop");
}
void loop() {
  if (running && millis() - started >= 700UL) {
    stopAll();
    Serial.println("Timed stop");
  }
  if (!Serial.available()) return;
  char c = Serial.read();
  if (c == '\n' || c == '\r') return;
  stopAll();
  int channel = -1;
  if (c == 'l') channel = 0;
  if (c == 'L') channel = 1;
  if (c == 'r') channel = 2;
  if (c == 'R') channel = 3;
  if (channel >= 0) {
    // Do not queue commands or increase duty to overcome a stall.
    pwm(channel, 100);
    started = millis();
    running = true;
    Serial.println("Brief motor pulse; x stops");
  } else Serial.println("Stopped");
}
```

Combined sensor diagnostic

File: tests/iphone/05_sensors_together/05_sensors_together.ino

Disconnect motor power again before uploading. Move the sonar target while slowly hand-turning each sensed wheel.

PASS: Distance and the correct edge count both respond; motor outputs remain off.

NEED HELP: If one fails, repeat its individual test and check supply loading and common ground. This is a diagnostic, not precision odometry.

```
// JARzLabs iPhone-ESP32 component test, documentation v1.1.0.
// Adult supervision. Disconnect motor power before uploading.
// Matches the dedicated openbot_ios firmware pin map.
#include <Arduino.h>
const int motorPins[] = {16, 17, 25, 26};
void holdMotorsOff() {
    for (int pin : motorPins) {
        pinMode(pin, OUTPUT);
        digitalWrite(pin, LOW);
    }
}
const int TRIG = 32, ECHO = 33;
void readSonar() {
    digitalWrite(TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    unsigned long us = pulseIn(ECHO, HIGH, 30000UL);
    if (us == 0) Serial.println("NO ECHO - check wiring/target");
    else {
        Serial.print("Distance cm: ");
        Serial.println(us * 0.0343f / 2.0f, 1);
    }
}
const int LEFT = 34, RIGHT = 35;
volatile unsigned long leftEdges = 0, rightEdges = 0;
void IRAM_ATTR leftChanged() { leftEdges++; }
void IRAM_ATTR rightChanged() { rightEdges++; }
void setupSpeed() {
    // ESP32 GPIO34/35 need external bias for open-collector outputs.
    pinMode(LEFT, INPUT);
    pinMode(RIGHT, INPUT);
    attachInterrupt(digitalPinToInterrupt(LEFT), leftChanged, CHANGE);
    attachInterrupt(digitalPinToInterrupt(RIGHT), rightChanged, CHANGE);
}
void readSpeed() {
    noInterrupts();
    unsigned long l = leftEdges, r = rightEdges;
    interrupts();
    Serial.print("L edges="); Serial.print(l);
    Serial.print(" R edges="); Serial.print(r);
    Serial.print(" | raw L="); Serial.print(digitalRead(LEFT));
    Serial.print(" R="); Serial.println(digitalRead(RIGHT));
}
void setup() {
    holdMotorsOff();
    Serial.begin(115200);
    pinMode(TRIG, OUTPUT);
    pinMode(ECHO, INPUT);
    setupSpeed();
}
void loop() {
    readSonar();
    readSpeed();
    delay(250);
}
```

iPhone BLE check without motors

File: tests/iphone/06_ble/06_ble.ino

Keep motor power off. Upload, then use a BLE scanner on iPhone, allow Bluetooth and scan for JarzRover Component Test. Connect, open service da8469b0-23dc-44a1-a230-5b36c2c6bb41 and read characteristic da8469b1-23dc-44a1-a230-5b36c2c6bb41.

PASS: The value is JARZLabs BLE OK - motors OFF. Disconnect the scanner; reset ESP32 and scan again to repeat.

NEED HELP: Use a BLE scanner, not the phone Bluetooth pairing list. This test uses a separate service and is not compatible with Robot control. Restore normal firmware before the app test.

```
// JARZLabs iPhone-ESP32 component test, documentation v1.1.0.
// Adult supervision. Disconnect motor power before uploading.
// Matches the dedicated openbot_ios firmware pin map.
#include <Arduino.h>
const int motorPins[] = {16, 17, 25, 26};
void holdMotorsOff() {
  for (int pin : motorPins) {
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
  }
}
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
void setup() {
  holdMotorsOff();
  Serial.begin(115200);
  BLEDevice::init("JarzRover Component Test");
  BLEServer *server = BLEDevice::createServer();
  BLEService *service = server->createService(
    "da8469b0-23dc-44a1-a230-5b36c2c6bb41");
  BLECharacteristic *value = service->createCharacteristic(
    "da8469b1-23dc-44a1-a230-5b36c2c6bb41",
    BLECharacteristic::PROPERTY_READ);
  value->setValue("JARZLabs BLE OK - motors OFF");
  service->start();
  BLEAdvertising *advertising = BLEDevice::getAdvertising();
  advertising->addServiceUUID(service->getUUID());
  advertising->setScanResponse(true);
  advertising->start();
  Serial.println("Scan with a BLE scanner; read test characteristic");
}
void loop() { delay(100); }
```

Restore normal firmware and test the apps

- 1 Switch off and disconnect motor power. Open firmware/openbot_ios/openbot_ios.ino from this ZIP with the same board selection.
- 2 Verify and upload. This restores OpenBot: DIY_ESP32 with the normal BLE service; component-test output is not the normal app protocol.
- 3 Follow [Installation Guide](#) for app installation and connection. Keep wheels raised while testing direction, stop on release and disconnect behavior. A successful component test does not establish a complete app-to-rover pass.
- 4 Record each test, expected/actual result, board identity and firmware revision. Do not mark PASS just because upload succeeded.

Contact Jarzlabs24@gmail.com for help. Include your kit version, board label, app version, test name and result. A parent or adult helper can send the request.

These sketches are checked by compilation and source comparison as recorded in the release verification notes. No new physical hardware test is claimed by this document.