Circuit Documentation for ESP32

Summary

This circuit manages precise servo motor movements using an Adafruit PCA9685 PWM Servo Breakout board, interfaced with an ESP32 microcontroller. The servos are powered by 18650 Li-ion batteries, with a rocker switch to manage the power supply. The Adafruit PCA9685 allows independent control of up to 16 servos through the I2C interface with the ESP32. The circuit features a capacitor for voltage smoothing and terminal PCBs for connecting the batteries and servos. The ESP32 handles interfacing with the servo control software, managing PWM signals for accurate positioning and coordination of the servos.

Microcontroller

- **ESP32**: Interfaces with the servo control software and manages PWM signals.

Power Supply

- **18650 Li-ion Batteries**: Rechargeable batteries providing the power source.
- **Rocker Switch**: Controls the power flow to the circuit.

Servo Control

- **Adafruit PCA9685 PWM Servo Breakout**: 16-channel, 12-bit PWM I2C-controlled servo driver for precise control of multiple servos.

Servos

- **Servos**: Actuators precisely controlled for position, receiving PWM signals from the Adafruit PCA9685.

Power Distribution

- **Terminal PCB 2 Pin**: Connectors for making quick and reversible connections of the power supply.

Passive Components

- **Electrolytic Capacitor**: 25V 100µF capacitor for voltage smoothing in the power supply.

Sensors

- **Ultrasonic Sensor (HC-SR04)**: For object detection.
- **MPU6050**: For acceleration and gyroscopic data.

Miscellaneous

- **Circular LED**: Connected to the ESP32 for visual indication.

Wiring Details

ESP32

- **5V**: → VCC of the Ultrasonic Sensor (HC-SR04)
- **GND**: → GND of all components
- **G12 (Trig)**: → Trig pin of the Ultrasonic Sensor (HC-SR04)
- **G14 (Echo)**: → Echo pin of the Ultrasonic Sensor (HC-SR04)
- **G13 (VCC)**: → VCC of the Circular LED
- **SDA**: → SDA pin of the Adafruit PCA9685 and MPU6050
- **SCL**: → SCL pin of the Adafruit PCA9685 and MPU6050

Adafruit PCA9685 PWM Servo Breakout

- **5.0V**: → VCC of the servos
- **GND**: → GND of the servos
- **SDA**: → SDA pin of the ESP32
- **SCL**: → SCL pin of the ESP32
- **PWM0 - PWM15**: → Pulse pin of corresponding servo

Servos

- **VCC**: → 5.0V pin of the Adafruit PCA9685
- **GND**: → GND pin of the Adafruit PCA9685
- **Pulse**: → Corresponding PWM pin on the Adafruit PCA9685

18650 Li-ion Batteries

- **+**: → Rocker switch and other batteries as required for desired voltage and capacity
- **-**: → Ground distribution via terminal PCBs

Rocker Switch

- **Input**: → Positive terminal of the battery pack
- **Output**: → Power distribution network and Adafruit PCA9685

Terminal PCB 2 Pin

- Connects the power supply to the Adafruit PCA9685 and servos.

Electrolytic Capacitor

- **+**: → Positive side of the power supply
- **-**: → Ground side of the power supply

Ultrasonic Sensor (HC-SR04)

- **VCC**: → 5V
- **Trig**: → G12 of ESP32
- **Echo**: → G14 of ESP32
- **GND**: → GND

MPU6050

- **VCC**: → 3.3V of ESP32
- **SDA**: → SDA of ESP32
- **SCL**: → SCL of ESP32

Circular LED

- **VCC**: → G13 of ESP32
- **GND**: → GND

Servo Pin Mapping

- **Pin 0 to 2**: Servos labeled as LH1, LH2, LH3
- **Pin 3 to 7**: Servos labeled as LL1, LL2, LL3, LL4, LL5
- **Pin 8 to 10**: Servos labeled as RH1, RH2, RH3
- **Pin 11 to 15**: Servos labeled as RL1, RL2, RL3, RL4, RL5

Documented Code

```
#include <Wire.h>
#include "thingProperties.h"            // this link us to cloud
#include <ESP32Servo.h>                 // Servo library
#include <NewPing.h>                    // Ultrasonic Sensor library
#include <Adafruit_MPU6050.h>           // for mpu 6050
#include <Adafruit_Sensor.h>



//.................................................. for servo movement
static const int servoPin = 23;
Servo myservo;
int minangle =40;
int maxangle =140;

//.................................................. for ultrasonic sensor
#define TRIGGER_PIN 12
#define ECHO_PIN 14
#define MAX_DISTANCE 100
int maxDistance = 60;
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
```

```
// ...................................................for arc reactor led
#define LED_PIN 13
int limit = 5;


//....................................................for mpu 6050
Adafruit_MPU6050 mpu;



void setup() {


  Serial.begin(9600);
  delay(1500);

  //                                                      [servo setup]
  myservo.attach(servoPin);

  //                                                      [led setup]
  pinMode(LED_PIN, OUTPUT);


  //                                                     [mpu 6050 setup]
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Connected Successfully!");

  mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);


  //                                                     [cloud setup]
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}


//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@


void controlLEDBrightness(int distance) {
  int brightness = 0;
  //if (distance > limit && distance <= maxDistance) {
  brightness = map(distance, limit , maxDistance, 255, 0);
  light = brightness;
  ArduinoCloud.update();
```

```
  analogWrite(LED_PIN, brightness);
  }

  /*else {
    digitalWrite(LED_PIN, LOW); // Turn off LED if no object detected
  }*/


//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@


void blinkLED(int times, int delayTime) {
  for (int i = 0; i < times; i++) {
    digitalWrite(LED_PIN, HIGH);
    delay(delayTime);
    digitalWrite(LED_PIN, LOW);
    delay(delayTime);
  }
}

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@


void moveHead() {
  static int position = minangle;
  static int increment = 3;
  bool objectDetected = false;
  myservo.write(position);
  servo = position;

  unsigned int distance = sonar.ping_cm();
  ultrasonic = distance;
  ArduinoCloud.update();

  // Led brightness

  if (distance == 0 || distance > maxDistance)
  {                                                       // when nothing detected
turn of led and save battery
     digitalWrite(LED_PIN, LOW);
  }


  if (distance > 1 && distance <= limit)
  {                                                        // for detection of very
close object it will blick led
    objectDetected = true;
    blinkLED(3 , 500);
  }


  if (distance > limit && distance <= maxDistance)
  {                                                        //for detection of person
```

```
nearby at given distance
    objectDetected = true;
    controlLEDBrightness(distance);
  }




  Serial.print("Servo position: ");
  Serial.print(position);
  Serial.print(" | Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (!objectDetected) {                                  // if detecs something no
incrementation value given to servo... i will stop rotating
    position += increment;
    if (position >= maxangle || position <= minangle ) {
      increment = -increment;
    }
  }

  delay(10);  // head rotation speed
}

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

void mpusensor() {

  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  acceX = a.acceleration.x;
  acceY = a.acceleration.y;

  ArduinoCloud.update();

  Serial.print(" | aX: "); Serial.print(acceX);
  Serial.print(" | aY: "); Serial.println(acceY);
  delay(100);

}




//
##############################################################################################
######################


void loop() {

  ArduinoCloud.update();
  moveHead();
```

```
  mpusensor();

}

//
##########################################################################################
####################


void onAcceXChange() {

}

void onServoChange() {

}

void onLightChange() {

}

void onUltrasonicChange() {

}

void onAcceYChange(){

}
```

thingproperties.h

cpp ```

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char DEVICE_LOGIN_NAME[]  = "fbc1b044-034e-4c66-8e48-ac5a190f918e";

const char SSID[]               = "Tracker_25";    // Network SSID (name)
const char PASS[]               = "nointernet";    // Network password (use for WPA,
or use as key for WEP)
const char DEVICE_KEY[]  = "SVFfhQERJ9eaWVKCsYaFvCKqI";    // Secret device password


void onServoChange();
void onLightChange();
void onUltrasonicChange();
void onAcceXChange();
void onAcceYChange();

int servo;
int light;
float ultrasonic;
```

```
float acceX;
float acceY;


void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(servo, READWRITE, ON_CHANGE, onServoChange);
  ArduinoCloud.addProperty(light, READWRITE, ON_CHANGE, onLightChange);
  ArduinoCloud.addProperty(ultrasonic, READ, ON_CHANGE, onUltrasonicChange);
  ArduinoCloud.addProperty(acceX, READWRITE, ON_CHANGE, onAcceXChange);
  ArduinoCloud.addProperty(acceY, READWRITE, ON_CHANGE, onAcceYChange);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```