Glove-Powered Electric Skateboard: Complete Build Guide

Comprehensive Parts List

Hardware Components

- Electric skateboard motor (Amazon link: https://a.co/d/9yW1uMH)
- Longboard deck (wood, to be custom cut)
- Skateboard trucks (standard longboard size)
- Skateboard wheels (compatible with custom pulley system)
- HTD 5M timing belt (30 teeth)
- Metal pulley for motor shaft
- DeWalt battery (18V or 20V recommended)
- DeWalt battery holder/adapter
- Electronic Speed Controller (ESC) - bidirectional type
- M3, M4, and M5 screws and nuts (assorted lengths)
- Thick wood sheet for CNC cutting (minimum 1/2" thick)

Electronics

- 3 × ESP32 XIAO microcontrollers
- Flex sensors (Amazon link: https://a.co/d/cbANXsr)
- Thin cotton glove
- Soldering wire
- Heat shrink tubing
- Jumper wires
- Breadboard (for prototyping)
- PCB (optional, for final build)
- 10kΩ resistors (for flex sensor voltage divider)

Tools Required

- 3D printer
- CNC machine (Shaper preferred)
- Soldering iron and solder
- Drill with assorted bits
- Screwdriver set
- Needle and thread
- Hot glue gun

- Superglue
- Multimeter
- Wire cutters/strippers
- Calipers or measuring tape
- Computer with Fusion 360 installed

Part 1: CAD Design and Fabrication

Motor Mount & Pulley System Design

1. **Measuring the Motor**
    - Carefully examine the motor you purchased from Amazon.
    - Using calipers, measure the diameter of the motor shaft (typically 8mm or 10mm for skateboard motors).
    - Measure the bolt pattern on the face of the motor where the mounting holes are located. Note both the bolt hole diameter and the pattern diameter.
    - Record all these measurements for use in CAD design.
2. **Designing the Motor Mount in Fusion 360**
    - Open Fusion 360 and create a new design.
    - Begin by creating a rectangular base plate (approximately 70mm × 50mm × 5mm) that will attach to the skateboard.
    - Create a perpendicular face (90° angle) extending from one edge of the base plate.
    - On this face, draw the hole pattern matching the motor's face pattern.
    - Add mounting holes on the base plate for attaching to the skateboard deck.
    - Include slots rather than fixed holes on the base plate to allow for belt tension adjustment.
    - Add reinforcement ribs between the base and perpendicular face for added strength.
    - Export the design as an STL file for 3D printing.
3. **Pulley System Design**
    - For the motor pulley:
        - If you already have a metal pulley for the motor shaft, measure its diameter and teeth count.
        - If designing one, create a cylinder matching the motor shaft diameter with a set screw hole.
        - Add HTD 5M tooth profile around the cylinder (use Fusion 360's gear generator or download a template).
    - For the wheel pulley:
        - Measure your skateboard wheel or download a model from online libraries.
        - Create a pulley disk with the same HTD 5M profile but with a larger diameter (gear ratio typically 1:3 to 1:4).

        - Add holes that align with the wheel's spokes.
        - Design the pulley to securely attach through the wheel's spokes with M3 or M4 bolts.
    - Calculating belt length and position:
        - Use the following formula to calculate the correct belt length:
          $L = 2C + \pi(D+d)/2 + (D-d)^2/4C$
          Where:
            - L = Belt length
            - C = Center distance between pulleys
            - D = Pitch diameter of larger pulley
            - d = Pitch diameter of smaller pulley
        - For proper belt tension, the center distance should allow for approximately 1/4 inch (6mm) of belt deflection when pressed with moderate force.
4. **Skateboard Deck Design**
    - Create a new sketch in Fusion 360 for the deck design.
    - Draw a longboard shape (approximately 900-1000mm length and 200-250mm width).
    - Round all corners with appropriate fillets.
    - Import the motor mount and truck designs to visualize placement.
    - Add mounting holes for:
        - Front and rear trucks
        - Motor mount
        - Battery holder
        - Electronics enclosure
    - Export the design as a DXF file for CNC cutting.
5. **Electronics Enclosure Design**
    - Create a rectangular box approximately 120mm × 80mm × 40mm.
    - Add internal mounting points for the ESP32 board and ESC.
    - Design a snug fit for all cables with strain relief.
    - Include ventilation holes or slots to prevent overheating.
    - Design a removable lid secured with screws for easy access.
    - Add mounting flanges with screw holes to attach to the skateboard.
    - Export as STL for 3D printing.
6. **Fabrication Process**
    - 3D print the motor mount, wheel pulley, and electronics enclosure using high-strength filament (PETG or ABS recommended).
    - Print with at least 50% infill for structural components.
    - For the deck:
        - Prepare your wood sheet (preferably 1/2" Baltic birch plywood or similar).
        - Cover the sheet with Shaper-machine tape.
        - Import the DXF file into the Shaper CNC software.
        - Follow the on-screen guidance to cut out the deck shape.
        - Sand all edges thoroughly after cutting.

- Drill all mounting holes according to your design specifications.
- Apply waterproof sealant or polyurethane to protect the wood.

Part 2: Glove Controller Assembly

Flex Sensor Integration

1. **Preparing the Flex Sensors**
   - Solder socket wires to each terminal of the flex sensors.
   - Use red wire for the power terminal and black for ground.
   - Apply heat shrink tubing to protect the solder joints.
   - Test each sensor with a multimeter to verify functionality.
2. **Glove Preparation**
   - Select a thin cotton glove that fits your hand comfortably.
   - Turn the glove inside out.
   - Mark the positions where the flex sensors will be attached (typically along the back of the index and middle fingers).
3. **Sewing Process**
   - Insert a popsicle stick into the finger where you'll be attaching the sensor to prevent sewing through both sides.
   - Thread a needle with strong thread and tie a knot by:
     - Wrapping the thread around your index finger twice.
     - Rolling the loops off your finger while pinching them.
     - Pulling the end of the thread through the loops to form a secure knot.
   - Position the flex sensor on the backside of the finger.
   - Begin sewing by entering from inside the glove and exiting near one edge of the sensor.
   - Create small stitches (5mm apart) that loop over the edges of the sensor without piercing it.
   - Continue sewing along both long edges of the sensor.
   - Once secure, apply a small amount of superglue at the top end of the sensor to anchor it firmly.
   - Repeat for each flex sensor you're installing.
4. **Wiring the Glove**
   - Route the wires along the back of the glove to the wrist area.
   - Secure the wires with small stitches or fabric glue.
   - Create a strain relief point at the wrist by sewing a small loop of excess wire.
   - Bundle all wires and add a connector for easy attachment to the microcontroller.

Part 3: Electronics and Programming

ESP32 Microcontroller Setup

1. **I2C and ESP-NOW Configuration**
    - Connect three ESP32 XIAO microcontrollers as follows:
        - Board 1: Interfaces with flex sensors (Sensor Board)
        - Board 2: Communicates with Board 1 via I2C and transmits via ESP-NOW (Transmitter Board)
        - Board 3: Receives ESP-NOW signals and controls the ESC (Receiver Board)

Wiring Diagram:

```
 Flex Sensor Board (ESP32 #1)
┌─────────────────┐
│                 │
│ Flex Sensors    │     I2C
│ (ADC2 pins)     ├──────────────▶ ESP32 #2 ┐
│                 │     SDA/SCL              │ ESP-NOW
└─────────────────┘                          ├──────────▶ ESP32 #3 ──▶
ESC ──▶ Motor
                                             │ Wireless

                                                  ┘
```

2.

3. **Sensor Board Setup (ESP32 #1)**
    - Connect the flex sensors to ADC2 pins using voltage dividers:
        - Connect one end of the flex sensor to 3.3V
        - Connect the other end to a 10kΩ resistor and the ADC input pin
        - Connect the other side of the resistor to GND
    - Wire the I2C connection:
        - Connect SDA to GPIO 21 on both boards
        - Connect SCL to GPIO 22 on both boards
        - Connect GND between boards

Code for Flex Sensor Board (ESP32 #1)

cpp
```cpp
#include <Wire.h>

// Define pins
const int flexPin1 = 32; // ADC2 pin
const int flexPin2 = 33; // ADC2 pin

// Variables for sensor values
int flexValue1 = 0;
int flexValue2 = 0;

void setup() {
  Serial.begin(115200);
```

```
  Wire.begin(0x08); // Initialize I2C as slave with address 0x08
  Wire.onRequest(requestEvent); // Register event
}

void loop() {
  // Read flex sensor values
  flexValue1 = analogRead(flexPin1);
  flexValue2 = analogRead(flexPin2);

  // Map flex sensor range to useful values (0-180)
  // Calibrate these values based on your specific flex sensors
  flexValue1 = map(flexValue1, 1000, 4095, 0, 180);
  flexValue2 = map(flexValue2, 1000, 4095, 0, 180);

  // Constrain values to prevent out-of-range issues
  flexValue1 = constrain(flexValue1, 0, 180);
  flexValue2 = constrain(flexValue2, 0, 180);

  Serial.print("Flex 1: ");
  Serial.print(flexValue1);
  Serial.print(", Flex 2: ");
  Serial.println(flexValue2);

  delay(50); // Small delay for stability
}

// Function called when master requests data
void requestEvent() {
  Wire.write(flexValue1);
  Wire.write(flexValue2);
```

4. }

Code for I2C Master and ESP-NOW Transmitter (ESP32 #2)

cpp

```
#include <Wire.h>
#include <esp_now.h>
#include <WiFi.h>

// Define receiver MAC address (replace with your ESP32 #3 MAC)
uint8_t receiverMacAddress[] = {0x24, 0x6F, 0x28, 0x12, 0x34, 0x56};

// Variables for sensor values
uint8_t flexValue1 = 0;
```

```
uint8_t flexValue2 = 0;

// Create data structure
typedef struct struct_message {
  uint8_t throttle;
  uint8_t brake;
} struct_message;

struct_message skateBoardData;

// Callback function for when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t
status) {
  Serial.print("Last Packet Send Status: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success"
: "Delivery Fail");
}

void setup() {
  Serial.begin(115200);

  // Initialize I2C as master
  Wire.begin();

  // Set up ESP-NOW
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register send callback
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, receiverMacAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
```

```
    return;
  }
}

void loop() {
  // Request 2 bytes from slave device
  Wire.requestFrom(0x08, 2);

  if (Wire.available() >= 2) {
    flexValue1 = Wire.read();
    flexValue2 = Wire.read();

    Serial.print("Received from I2C - Flex 1: ");
    Serial.print(flexValue1);
    Serial.print(", Flex 2: ");
    Serial.println(flexValue2);

    // Prepare data to send
    skateBoardData.throttle = flexValue1;
    skateBoardData.brake = flexValue2;

    // Send data via ESP-NOW
    esp_err_t result = esp_now_send(receiverMacAddress, (uint8_t *)
&skateBoardData, sizeof(skateBoardData));

    if (result == ESP_OK) {
      Serial.println("Sent with success");
    } else {
      Serial.println("Error sending the data");
    }
  }

  delay(100); // Small delay between transmissions
```

5. }

Code for ESP-NOW Receiver and ESC Controller (ESP32 #3)

cpp

```
#include <esp_now.h>
#include <WiFi.h>
#include <ESP32Servo.h>

// Define ESC control pin
const int escPin = 13;
```

```
// Create ESC control object
Servo ESC;

// Variables for throttle control
int throttleValue = 0;
int brakeValue = 0;
int escSignal = 1500; // Neutral point for bidirectional ESC is 1500
microseconds

// Create data structure to receive
typedef struct struct_message {
  uint8_t throttle;
  uint8_t brake;
} struct_message;

struct_message incomingData;

// Callback function executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData_buf,
int len) {
  memcpy(&incomingData, incomingData_buf, sizeof(incomingData));

  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("Throttle: ");
  Serial.println(incomingData.throttle);
  Serial.print("Brake: ");
  Serial.println(incomingData.brake);

  // Update control values
  throttleValue = incomingData.throttle;
  brakeValue = incomingData.brake;
}

void setup() {
  Serial.begin(115200);

  // Set device as WiFi Station
  WiFi.mode(WIFI_STA);

  // Initialize ESC control
  ESC.attach(escPin, 1000, 2000); // 1000-2000 microsecond range for
standard ESC
```

```
  ESC.writeMicroseconds(1500); // Set to neutral position (VERY
IMPORTANT)
  delay(3000); // Give ESC time to initialize

  // Initialize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register callback function
  esp_now_register_recv_cb(OnDataRecv);

  Serial.println("Receiver Ready");
}

void loop() {
  // Calculate ESC signal
  // IMPORTANT: Bidirectional ESCs use 1500µs as neutral point (NOT
0µs)
  // Values below 1500 are reverse, values above are forward

  // Map throttle from 0-180 to appropriate ESC range
  // Assuming flexing forward = accelerate, flexing back = brake
  if (throttleValue > 90) { // Forward motion
    escSignal = map(throttleValue, 90, 180, 1500, 2000);
  } else if (brakeValue > 90) { // Braking/reverse
    escSignal = map(brakeValue, 90, 180, 1500, 1000);
  } else {
    escSignal = 1500; // Neutral position
  }

  // Send the signal to the ESC
  ESC.writeMicroseconds(escSignal);

  Serial.print("ESC Signal: ");
  Serial.println(escSignal);

  delay(15); // Small delay for stability
}

// ESC Calibration Function (call this from setup if needed)
void calibrateESC() {
  Serial.println("ESC Calibration Starting");
  delay(1000);
```

```
Serial.println("Setting maximum throttle");
ESC.writeMicroseconds(2000);
delay(5000);
Serial.println("Setting minimum throttle");
ESC.writeMicroseconds(1000);
delay(5000);
Serial.println("Setting neutral throttle");
ESC.writeMicroseconds(1500);
delay(5000);
Serial.println("ESC Calibrated!");
```

6. }

Part 4: Final Assembly

Putting Everything Together

1. **Mount the Motor and Trucks**
   - Attach the trucks to the skateboard deck using appropriate screws.
   - Mount the motor mount to the skateboard deck using the predrilled holes.
   - Attach the motor to the motor mount.
   - Install the motor pulley onto the motor shaft, securing with set screws.
   - Attach the wheel pulley to the rear truck wheel, securing through the spokes.
   - Place the timing belt around both pulleys.
   - Adjust the motor mount position to achieve proper belt tension.
   - Tighten all screws securely.
2. **Install Battery Mount**
   - Drill two mounting holes on the bottom of the skateboard for the DeWalt battery holder.
   - Attach the battery holder using appropriate screws and lock nuts.
   - Connect the battery holder output wires to the ESC power input.
   - Ensure proper polarity and secure all connections.
3. **Install Electronics Box**
   - Mount the 3D-printed electronics box to the bottom of the skateboard.
   - Install the ESC and receiver ESP32 inside the box.
   - Connect the ESC signal wire to the ESP32 control pin.
   - Connect the ESC power wires to the battery and motor.
   - Secure all connections and arrange wires neatly inside the box.
   - Attach the box lid with screws.
4. **Final Glove Setup**
   - Install the flex sensor ESP32 and the transmitter ESP32 in a small enclosure.
   - Attach this enclosure to the wrist area of the glove.
   - Connect the flex sensors to the ESP32 board.

    - Power the ESP32s with a small LiPo battery.
    - Secure all components to prevent movement while in use.
5. **Testing and Calibration**
    - Insert a fully charged DeWalt battery into the holder.
    - Power on the glove controller.
    - Verify that the flex sensors are reading correctly.
    - Test the motor response to glove movements.
    - Adjust the code parameters if necessary for smoother control.
    - Test acceleration and braking in a safe, open area.
6. **Safety Check**
    - Ensure all screws and connections are tight.
    - Check that the timing belt has proper tension.
    - Verify that the ESC and motor are not overheating during operation.
    - Test the emergency stop function (releasing all flex sensors).
    - Wear appropriate safety gear (helmet, pads) during operation.

Troubleshooting Tips

- **Motor doesn't respond**: Check ESP-NOW connection, verify ESC calibration.
- **Inconsistent throttle control**: Recalibrate flex sensor mapping values in code.
- **Belt slipping**: Increase belt tension or check for worn teeth.
- **Short battery life**: Verify there are no shorts in the system, consider a higher capacity battery.
- **Overheating components**: Add additional ventilation to the electronics enclosure.

Maintenance

- Regularly check belt tension and condition.
- Inspect all mounting screws and tighten as needed.
- Clean bearings and mechanical components.
- Update firmware as improvements are made.
- Replace flex sensors if sensitivity decreases over time.