ESP32 USE JOYSTICK TO FADE (PWM) AN LED. WHICH WE WILL LATER HOOK TO A SMALL (5A) BRUSHED MOTOR CONTROLLER TO DRIVE A YELLOW GEARED MOTOR. ENUMERATE THE PINS AND WRITE THE CODE

Jun 13, 1:40 AM

I can certainly help you with that! You'll need to set up your ESP32 to read analog input from the joystick and then use that value to control the brightness of an LED via PWM. This setup will be easily adaptable for a brushed motor controller later.

Here's an enumeration of the pins and the Arduino (C++) code for your ESP32.

### **Explanation and Next Steps:**

**1. Pin Enumeration:**

* **Joystick X-axis (JOYSTICK\_X\_PIN = 34):** GPIO 34 is an Analog-to-Digital Converter (ADC) input pin on the ESP32. This is where you'll connect the "VRx" output of your joystick module.
* **Joystick Y-axis (JOYSTICK\_Y\_PIN = 35):** GPIO 35 is another ADC input pin. You can connect the "VRy" output here. While this example primarily uses the X-axis for LED control, having both defined is good for future expansion (e.g., controlling two motors or directions).
* **LED Pin (LED\_PIN = 2):** GPIO 2 is a versatile pin on the ESP32 that supports PWM. You'll connect your LED (with a current-limiting resistor, typically 220-330 ohms) to this pin.

**2. How the Code Works:**

* **setup() function:**
  + Initializes serial communication for debugging output.
  + Configures the ESP32's built-in LED Control (LEDC) peripheral for PWM. We define a PWM\_CHANNEL (0), a PWM\_FREQUENCY (5000 Hz, a good balance for flicker-free LED control), and a PWM\_RESOLUTION (8 bits, meaning the brightness can be set from 0 to 255).
  + Attaches the LED\_PIN to this configured PWM channel.
* **loop() function:**
  + analogRead(JOYSTICK\_X\_PIN): Reads the analog voltage from the joystick's X-axis. The ESP32's ADC has a default resolution of 12 bits, meaning it will return a value between 0 (for 0V) and 4095 (for 3.3V, assuming your joystick is powered by 3.3V).
  + map(joystickValueX, 0, 4095, 0, 255): This crucial function scales the input joystickValueX (which ranges from 0 to 4095) to the desired PWM duty cycle range (0 to 255). When the joystick is at one extreme, pwmDutyCycle will be 0 (LED off), and at the other extreme, it will be 255 (LED full brightness).
  + ledcWrite(PWM\_CHANNEL, pwmDutyCycle): This command sets the PWM duty cycle for the LED, effectively controlling its brightness.
  + Serial.print(...): Prints the raw joystick value and the calculated PWM value to the Serial Monitor, which is very helpful for testing and debugging.
  + delay(20): A small delay to keep the Serial output readable and prevent the loop from running too fast.

**To use this code:**

1. **Hardware Connections:**
   * **Joystick:**
     + VCC -> ESP32 3.3V
     + GND -> ESP32 GND
     + VRx -> ESP32 GPIO 34
     + VRy -> ESP32 GPIO 35
     + SW (Switch) -> (Optional, connect to any digital input pin, e.g., GPIO 27, and pull-up resistor if needed)
   * **LED:**
     + Longer leg (Anode) -> Current-limiting resistor (e.g., 220 Ohm) -> ESP32 GPIO 2
     + Shorter leg (Cathode) -> ESP32 GND
2. **Software:**
   * Open the Arduino IDE.
   * Ensure you have the ESP32 board definitions installed.
   * Copy and paste the code into a new sketch.
   * Select your ESP32 board and the correct COM port.
   * Upload the code.
   * Open the Serial Monitor (at 115200 baud) to see the values.

This setup gives you a solid foundation for your project!