**Raspberry Pi Joystick Reader Using MCP3008**

1. **Introduction**

This document describes the functionality of a Python script designed to read joystick movements through the MCP3008 Analog-to-Digital Converter (ADC) connected to a Raspberry Pi. The script normalizes these readings for consistent output, regardless of the hardware setup.

1. **Setup Instructions**

**Hardware Setup:**

The required components are as follows:

* Raspberry Pi
* MCP3008 ADC
* Joystick module
* Breadboard and jumper wires

**Connections:**

1. Connect the MCP3008 VDD and VREF to the Raspberry Pi 3.3V.

2. Connect the MCP3008 AGND and DGND to the Raspberry Pi GND.

3. Connect the MCP3008 CLK, DOUT, DIN, and CS/SHDN to specific GPIO pins on the Raspberry Pi for SPI communication.

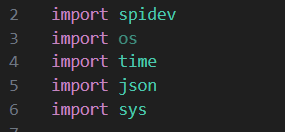
4. Connect the joystick's horizontal and vertical outputs to the MCP3008 channels (e.g., CH0 and CH1).

**Software Requirements:**

* Python 3.x installed on Raspberry Pi
* SPI interface enabled on Raspberry Pi (use raspi-config to enable SPI under "Interface Options")
* spidev Python library installed (sudo pip3 install spidev)

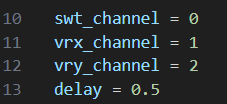
1. **Script Breakdown**

**Import Statements**

****

* **spidev**: For SPI communication with MCP3008.
* **time**: For handling delays and timing in the script.
* **json**: For saving and loading calibration data to/from a file.
* **sys**: For accessing command-line arguments.

**Global Variables**

****

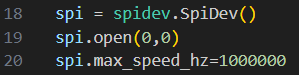
* **“swt\_channel”**, “**vrx\_channel”**, “**vry\_channel”:**

Define the MCP3008 channels connected to different parts of the joystick.

* **“Delay”:**

Defines the time interval (in seconds) between successive readings.

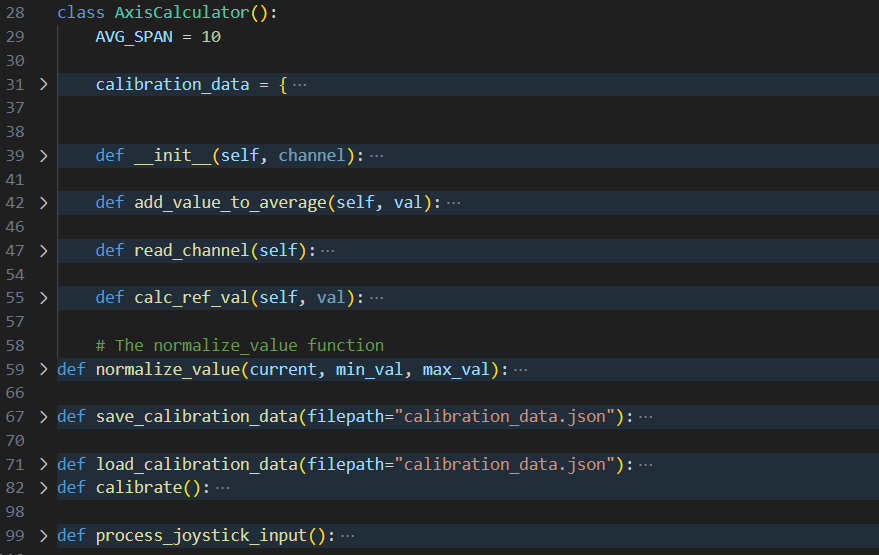
**SPI Setup**

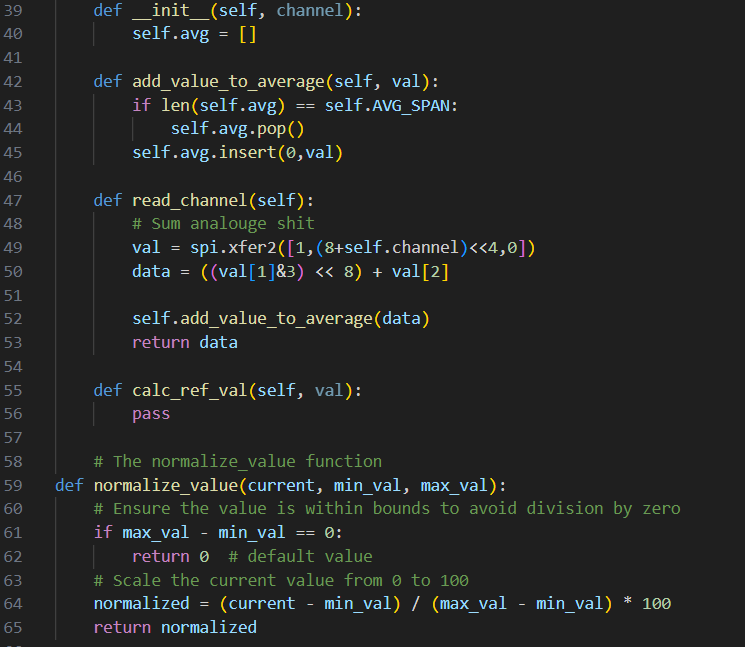
****

Initializes the SPI connection using “spidev”, setting up the speed and mode for communication with MCP3008.

**Class Definition (The”AxisCalculator” class)**

Handles the calculation of moving averages, normalization of joystick readings, and storage of calibration data.

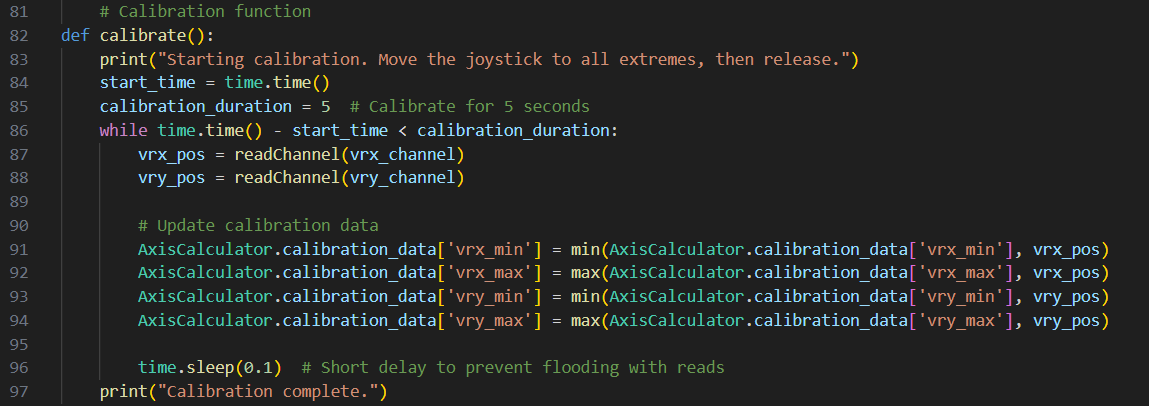




1. **Calibration Process**

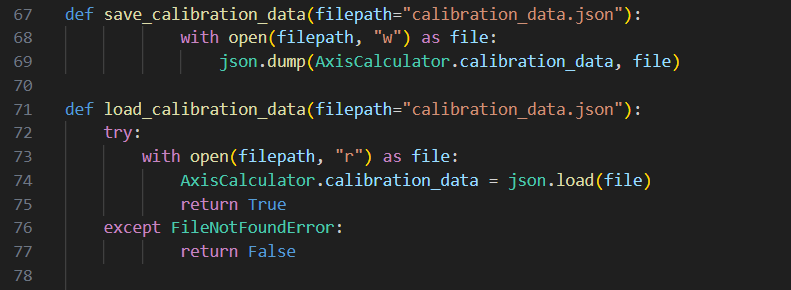
**Calibration Function (“calibrate”)**

Captures the maximum and minimum readings from the joystick during a user-induced calibration phase, ensuring that joystick movements are accurately reflected in normalized values.



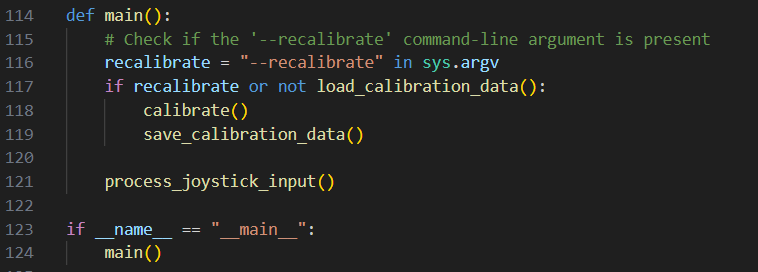
**Saving and Loading Calibration Data**

* **Saving:** Calibration data is saved to “**calibration\_data.json**”, allowing future sessions to use the same calibration without recalibration.
* **Loading:** Attempts to load existing calibration data at startup to bypass recalibration.



1. **Main Program Flow**

Explains how the script checks for the **--recalibrate** argument to decide whether to use existing calibration data or start a new calibration process.



1. **Running the Script**

* **Normal start:** “python3 /home/pi/Desktop/mcp.py”
* **Force recalibration:** “python3 /home/pi/Desktop/mcp.py --recalibrate”

1. **Expected Outputs**

**Console Outputs:**

Displays the normalized X and Y positions of the joystick and the state of the switch (if applicable), updated at intervals defined by “delay”.

