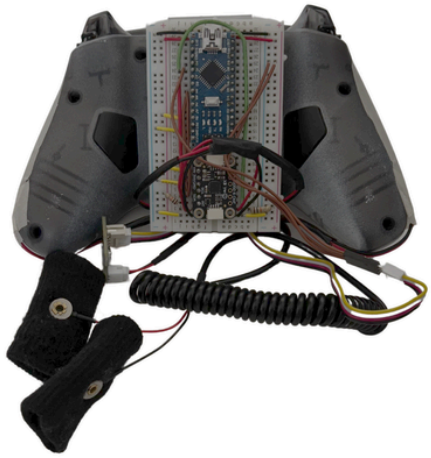
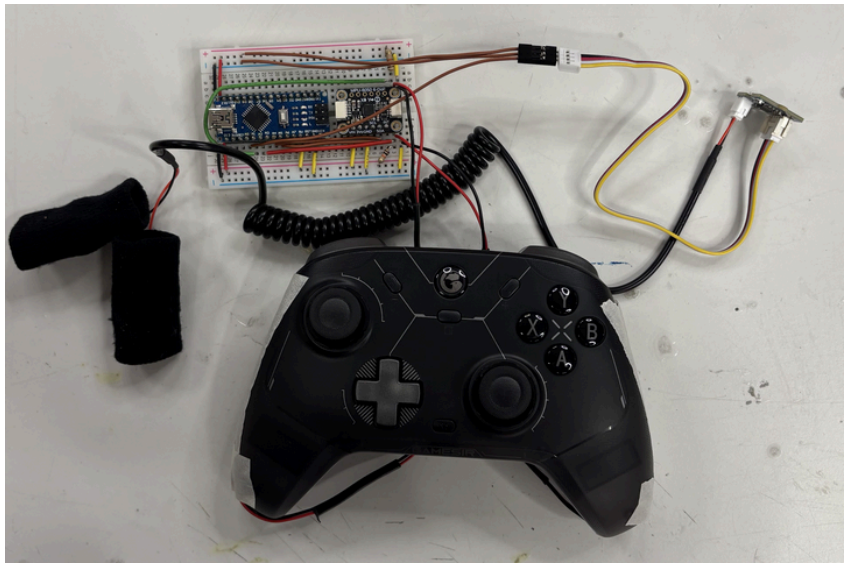
AUTODESK Instructables

Customisable Game Controller With Adaptive Sensors Build Guide

By ramyadnan in CircuitsArduino
Published Apr 27th, 2025

Introduction: Customisable Game Controller With Adaptive Sensors Build Guide





This tutorial will guide you through the process of creating a customisable game controller that integrates multiple sensors to track hand-grip pressure, motion, and emotional responses. The project uses a flex sensor, an accelerometer, and a galvanic skin response (GSR) sensor, allowing for dynamic interaction based on the player's physical and emotional state. Follow along to learn how to select components, assemble the controller, and get started with your own adaptable gaming setup.

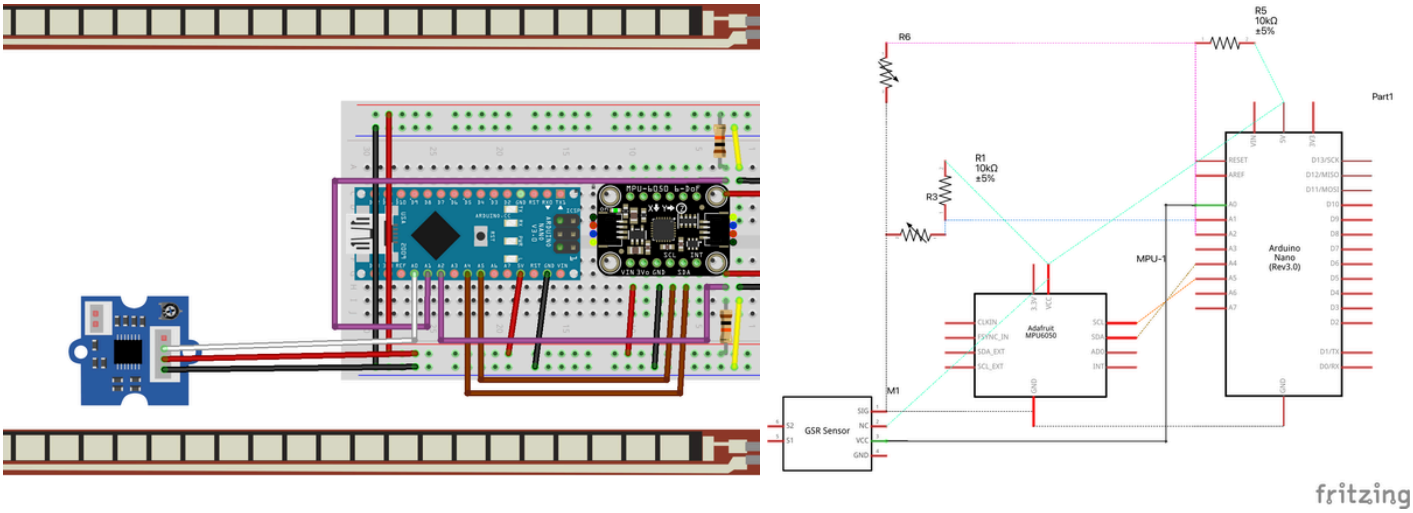
Supplies

For this tutorial you will need:

1. A game controller: For this guide I will be using GameSir Cyclone 2. However, any compatible game controller will work. When choosing a game controller, it is recommended to select one that has an STL file for the shell available online. This will make the 3D printing process much easier, and you can modify the shape of the shell, especially the back, to fit a breadboard.
2. Arduino Nano
3. Mini breadboard
4. 10K Ohm Resistor (x2)
5. Flex sensor (x2)
6. Grove GSR with finger sensor
7. MPU6050 6-DoF Accelerometer and Gyro
8. Access to a 3D printer: For customizing the controller’s housing to fit the components. A 3D printer will be used to print a shell for the controller, but you can also modify the shape of the shell's back to accommodate a breadboard.

In this project, I will be using a clipper to hold the breadboard securely in place, making it easily removable. This approach eliminates the need for permanently modifying the controller's shell, providing flexibility for future adjustments.

Step 1: The Circuit



The connections are straightforward, and you can refer to the image above for the breadboard circuit schematic. Just ensure all components are properly connected as shown, and you're good to go!

Step 2: The Clipper

This step is optional but highly recommended if you have access to a 3D printer. You can 3D print a clipper with a breadboard holder to securely hold the breadboard in place while still allowing it to be removable. This prevents the breadboard from being permanently attached to the back of your controller, offering flexibility for future adjustments or experimentation.

Attached is the STL file for the clipper with a breadboard holder that fits this particular game controller model uses in this project.

Step 3: The Arduino Code

The Arduino code is used to read sensor data and send it over the serial connection to the Python script. Below is the code that collects data from the flex sensor, accelerometer, and GSR sensor and sends it to the computer via serial communication.

Before you start, you'll need to install the **Adafruit_MPU6050** library, which you can access here. This library is essential for communicating with the accelerometer.

Once everything is wired up correctly, you can upload the following code to your Arduino:

Step 4: The Python Code

The Python code interfaces with the Arduino via serial communication to collect the sensor data in real-time. The data is logged to CSV files for further analysis. The script also includes simple keypress controls to start and stop logging.

The Python script listens for incoming data from the Arduino, logs it into a CSV file, and allows you to control the logging process using the 's', 'e', and 'q' keys.

Steps to Run the Python Code:

1. Ensure that the Arduino is connected to the correct serial port. Update the arduino_port in the Python script accordingly (e.g., /dev/cu.usbserial-1410 for macOS or COMx for Windows).
2. Run the Python script in your preferred IDE, such as Visual Studio Code.
3. Press **'s'** to start logging, **'e'** to stop logging, and **'q'** to quit the program.

The script will automatically create a new CSV file for logging and save the sensor data along with a timestamp. Below is the Python code:

Step 5: The Joystick Event Polling

In this part of the tutorial, we will be using C++ alongside the SDL3 package to handle joystick event polling. This will allow us to capture joystick inputs in real-time, making the interaction with the game controller responsive.

Step 1: Create a C++ Environment

To get started, you'll need to set up a C++ development environment. You can use any IDE or text editor you're comfortable with, but make sure you have C++11 or higher enabled in your project settings.

Step 2: Install SDL3

Next, you'll need to install the SDL3 package. You can download the SDL3 package from the official website here. Follow the instructions on the website to install the library for your operating system.

Step 3: Replace the main.cpp

Once SDL3 is installed, you'll need to replace the content of your main.cpp file with the code that handles joystick event polling. Below is an example of the C++ code using SDL3 to poll joystick events:

Step 6: The Data Visualisation

To visualise the raw data from the sensors, you can use the following Python code. This will allow you to plot and explore the data for further analysis. Feel free to modify the code to suit your specific needs and gain insights into your sensor data.

This code will display three plots:

1. Flex Sensor data over time.
2. GSR (Galvanic Skin Response) data.
3. Accelerometer and gyroscope data.

This project aims to assist those in Human-Computer Interaction (HCI), particularly in the gaming industry, by creating a reliable device to measure player analytics. By capturing and analysing sensor data, developers and researchers can gain insights into user interactions and emotional responses, enabling them to design better gaming experiences.

Step 7: Well Done!

Congratulations on completing the project! You've successfully built a game controller with adaptive sensors that provide valuable real-time data. I hope this project is useful and inspires further innovation in measuring player interaction and emotions in gaming.

GitHub Repository

For the complete project files, additional code, and updates, check out the GitHub repository for this project here:

GitHub Repository