Detailed Code Explanations:

1. `jiosavan.py` – Browser Control via Selenium

Purpose:
Launches Chrome, navigates to JioSaavn, and exposes functions to play/pause, skip tracks, mute, and adjust volume by clicking on the page's buttons.

a. Configuration

```
JIO_SAAVN_URL       = "https://www.jiosaavn.com/"
PLAYER_CONTROLS     = "#player > div.c-player__panel > 
ul.c-player__controls…"
PLAY_PAUSE_BTN      = PLAYER_CONTROLS + " > li:nth-child(3)"
NEXT_BTN            = PLAYER_CONTROLS + " > li.c-player__btn-next"
PREV_BTN            = PLAYER_CONTROLS + " > li.c-player__btn-prev"
```

- **`JIO_SAAVN_URL`** is the site URL.
- **`PLAYER_CONTROLS`** is the common prefix for the player's control toolbar.
- The other three constants build on that to point exactly at Play/Pause, Next, and Previous buttons via CSS selectors.

b. Driver Initialization

```
def init_driver():
    driver = webdriver.Chrome()
    driver.get(JIO_SAAVN_URL)
    time.sleep(5)  # give the page time to load the player
    return driver
```

- Starts a new Chrome window (requires `chromedriver` on your PATH).
- Waits 5 s to ensure the player UI is visible.

c. Click Helpers

```
def click_play_pause(driver):
    driver.find_element(By.CSS_SELECTOR, PLAY_PAUSE_BTN).click()

def click_next(driver):
    driver.find_element(By.CSS_SELECTOR, NEXT_BTN).click()

def click_previous(driver):
    driver.find_element(By.CSS_SELECTOR, PREV_BTN).click()
```

- Each finds the corresponding element by its CSS selector and fires a `.click()`.

d. Mute & Volume

```
def click_mute(driver):
    # toggles mute by clicking the volume icon (selector omitted
here)

def change_volume(driver, delta):
    # fetch the current volume slider position, adjust by delta,
then apply
```

- **`click_mute`** simply clicks the mute/unmute icon.
- **`change_volume`** reads the page's volume slider, adds or subtracts a small `delta` (e.g. +0.1 or –0.1), and updates the slider.

e. User Menu & Main Loop

```
def display_menu():
    print("1) Play/Pause\n2) Next\n3) Prev\n4) Mute\n5) Vol Up\n6)
Vol Down\n7) Exit")

def main():
    driver = init_driver()
    try:
        while True:
            display_menu()
            choice = input("Enter choice (1–7): ")
            # map "1" → click_play_pause, "2" → click_next, etc.
            if choice == "1":
                click_play_pause(driver)
            elif choice == "2":
                click_next(driver)
            # …
            elif choice == "7":
                break
            time.sleep(0.2)      # tiny debounce so clicks don’t
fire too rapidly
    finally:
        driver.quit()
```

- Prints a simple console menu.
- Reads your keystroke, calls the associated helper, then loops.
- On exit it cleanly quits Chrome.

---

2. `jiosavan_controller.py` – Serial Bridge between Micro:bit & Browser

Purpose:
Listens on a serial port for text commands coming from the Micro:bit and forwards them to `jiosavan.py`’s click/volume functions.

a. Setup

```
import serial
from jiosavan import init_driver, click_button, change_volume,
PLAY_PAUSE_BTN, NEXT_BTN, PREV_BTN

SERIAL_PORT = 'COM9'
BAUD_RATE   = 115200
```

- Uses `pyserial` to open `COM9` (adjustable) at 115 200 baud.

b. Main Loop

```
def main():
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
    driver = init_driver()

    try:
        while True:
            if ser.in_waiting:
                line = ser.readline().decode().strip()
                # e.g. line == 'PLAY', 'NEXT', 'PREV', 'MUTE', 'VU',
'VD'
                if line == 'PLAY':
                    click_button(driver, PLAY_PAUSE_BTN)
                elif line == 'NEXT':
                    click_button(driver, NEXT_BTN)
                elif line == 'PREV':
                    click_button(driver, PREV_BTN)
                elif line == 'MUTE':
                    click_mute(driver)
                elif line == 'VU':
                    change_volume(driver, +0.1)
                elif line == 'VD':
                    change_volume(driver, -0.1)
            time.sleep(0.1)
    except KeyboardInterrupt:
        print("Exiting…")
    finally:
        driver.quit()
        ser.close()
```

- **Reads** a line of text from the Micro:bit (e.g. `NEXT\n`).
- **Dispatches** it immediately to the corresponding web-control function.
- Handles clean-up on Ctrl+C.

---

3. `microbit.py` – Gesture & Button Logic on the Micro:bit

Purpose:
Runs on the BBC Micro:bit, watches for shakes and button presses, and sends simple text commands over USB UART.

a. Initialization

```
from microbit import display, accelerometer, button_a, button_b,
uart, running_time, sleep, Image
uart.init(baudrate=115200)
```

- Brings in `accelerometer`, `button_a`, `button_b`, and `uart`.
- Starts UART at 115 200 baud to match the PC side.

b. Detecting Events

```python
# Track when last A/B was pressed for double-press logic
last_a_time = last_b_time = 0
a_count = b_count = 0
DOUBLE_WINDOW = 2000  # ms

while True:
    now = running_time()

    # 1) Shake → Play/Pause
    if accelerometer.was_gesture('shake'):
        uart.write('PLAY\n')
        display.show(Image.PLAY)
        sleep(500)
        display.clear()

    # 2) Double A → Previous track
    if button_a.is_pressed():
        if now - last_a_time < DOUBLE_WINDOW:
            a_count += 1
        else:
            a_count = 1
        last_a_time = now

    if a_count == 2:
        uart.write('PREV\n')
        display.show(Image.ARROW_W)
        sleep(500)
        display.clear()
        a_count = 0

    # 3) Double B → Next track
    if button_b.is_pressed():
        if now - last_b_time < DOUBLE_WINDOW:
            b_count += 1
        else:
            b_count = 1
        last_b_time = now

    if b_count == 2:
        uart.write('NEXT\n')
        display.show(Image.ARROW_E)
        sleep(500)
        display.clear()
        b_count = 0

    # Handle single presses if no second press arrives in time…
    if a_count == 1 and now - last_a_time > DOUBLE_WINDOW:
        # treat single A as “Prev” as well
        uart.write('PREV\n')
        display.show(Image.ARROW_W)
        sleep(500)
        display.clear()
        a_count = 0

    if b_count == 1 and now - last_b_time > DOUBLE_WINDOW:
        # treat single B as “Next”
        uart.write('NEXT\n')
        display.show(Image.ARROW_E)
        sleep(500)
        display.clear()
        b_count = 0

    sleep(100)
```

1. **Shake gesture** → send `PLAY` (toggles play/pause).
2. **Double-tap A** within 2 s → send `PREV`.
3. **Double-tap B** within 2 s → send `NEXT`.
4. If only one tap of A/B happens and 2 s pass without a second tap, it still fires the same command (so you don’t have to tap twice if you just want prev/next).

- Each command ends with a newline, e.g. `"NEXT\n"`, which the PC side reads.
- A small icon briefly displays on the Micro:bit’s LED grid so you know your action was sent.

Putting It All Together

1. **On your PC**, run `jiosavan_controller.py`.
2. That launches Chrome and opens JioSaavn.
3. **On the Micro:bit**, run `microbit.py`.
4. When you **shake** the Micro:bit, it sends `PLAY` → Selenium clicks Play/Pause.
5. When you **double-press A/B**, it sends `PREV`/`NEXT` → Selenium skips back or forward.
6. You can extend the same pattern for volume up/down or mute with additional gestures or button combos.

Now each piece is modular:

- **Micro:bit** decides *when* to send commands.
- **Controller** forwards the text commands to the browser.
- **Selenium script** does the actual clicking on jiosaavn.com.