EGC 121 / Computer Architecture

JioSaavn Music Controller

Micro:Bit Project Report

Team Members:

- Ayush Patel (BT2024054)
- Kabir Ahuja (BT2024004)
- Kanav Kumar (BT2024021)
- Dayal Gupta (BT2024167)
- Parth Malhotra (BT2024197)
- Sachin Nain (BT2024203)
- Tanmay Dixit (BT2024016)

Introduction

This project implements a **micro:bit v2**–based controller for the **JioSaavn web music player**. The micro:bit reads **button presses** and **gesture inputs** and sends corresponding commands via a USB serial link to a Python program on the PC. The PC program uses **Selenium WebDriver** to automate the JioSaavn web page, executing **play/pause, next/previous track and volume up/down actions** in response to the micro:bit commands. The result is a responsive system allowing physical control of online music playback.

This work demonstrates a successful **integration of a microcontroller with web automation,** bridging physical inputs and a modern web service.

Objectives

- **Functional:** Build a controller that manages basic playback functions (play/pause, next/previous track, volume control) on the JioSaavn web player using a micro:bit.

- **Performance:** Ensure the system is responsive, with each command (button press or gesture) resulting in the action being performed on the player within approximately one second.

- **Validation:** Verify functionality by manually observing that each micro:bit input (e.g. button press) causes the JioSaavn player to respond correctly (such as changing tracks or adjusting volume).

Video Recording

A **demonstration** of the working project has been recorded and uploaded to Google Drive. The video showcases the micro:bit detecting **various button presses and gestures** and the **corresponding playback actions** being executed on the **JioSaavn web interface.**

Watch here:

https://drive.google.com/file/d/17tipGkDq-_PeUA8daU7xiVDqFFfhpIk-/view?usp=share_link

Micro:Bit Functionality

The following table summarizes the physical interactions supported by the micro:bit and the corresponding control actions performed on the JioSaavn web player. Each action was carefully mapped to a distinct button press or gesture to provide intuitive control over music playback:

Micro:bit Input	JioSaavn Action
Right Button (Button B) Pressed Once	Skip to the next track
Left Button (Button A) Pressed Once	Restart the current track
Left Button (Button A) Pressed After Restart	Go to the previous track
Right Button (Button B) Pressed Twice	Increase the volume
Left Button (Button A) Pressed Twice	Decrease the volume
Shake Gesture	Toggle Play/Paus

Implementation Details

- **Hardware:** We used only a BBC micro:bit v2 board. The micro:bit supports Python via MicroPython (microbit-micropython.readthedocs.io) and has built-in buttons, LEDs and motion sensors. It is connected to the PC via USB.. This allows the micro:bit to transmit single-character commands to the computer when the user presses buttons or moves the board.

- **Software:** The micro:bit runs a MicroPython script (`microbit.py`) that reads button A/B presses and gesture inputs and sends corresponding command codes (for play/pause, next, previous, volume up/down) over the USB serial link. On the PC, we used Python3 with Selenium WebDriver to control the Chrome browser. Selenium (Python bindings) automates web browser interactions (pypi.org) and ChromeDriver provides the WebDriver interface for Google Chrome (developer.chrome.com). The core software modules are:

    - *microbit.py:* Runs on the micro:bit; reads user inputs (buttons and gestures) and uses the serial API to send command characters to the PC.

    - *usb_controller.py:* Runs on the PC; listens on the USB serial port for incoming commands from the micro:bit. When a valid command is received, it calls the corresponding function in the browser control module.

    - *control_saavan.py:* Contains functions that use Selenium to control the JioSaavn web player. These functions find the relevant buttons or keys on the webpage and invoke play/pause, next track, previous track and volume adjustments when triggered by the micro:bit commands.

Code Explanation

- **jiosavan.py:**

This script uses Selenium to automate JioSaavn in Chrome. It defines functions for Play/Pause, Next, Previous, Mute, and Volume slider controls, initializes a WebDriver (with a 5 s load delay), and provides helper functions like `click_play_pause()`, `click_next()`, `click_previous()`, `click_mute()` and `change_volume()`—that locate and click or adjust the corresponding elements. A simple console menu loops to read user input (1–7) and dispatches to the appropriate helper, then cleanly quits Chrome on exit.

- **jiosavan_controller.py:**

Acting as a serial bridge, this script opens a `pyserial` connection (e.g. COM9 at 115 200 baud) and reuses the Selenium helpers from `jiosavan.py`. In its main loop it reads lines like `'PLAY'`, `'NEXT'`, `'PREV'`, `'MUTE'`, `'VU'`, and `'VD'` from the Micro:bit over UART and immediately calls `click_play_pause()`, `click_next()`, etc., or `change_volume(driver, ±0.1)`. On `KeyboardInterrupt` it quits the browser and closes the serial port.

- **microbit.py:**

Running on the BBC Micro:bit, this script initializes UART at 115 200 baud and monitors gestures and buttons:

- **Shake** → sends `"PLAY\n"` (play/pause)
- **Double-press A** → `"PREV\n"` (previous track)
- **Double-press B** → `"NEXT\n"` (next track)
  A single A/B tap also triggers the same command after a 2 s window. Each action displays a brief LED icon feedback, and commands end with `\n` so the controller script can parse them.

Challenges & Solutions

- **Double-Press Detection:** Distinguishing single vs. double button presses was challenging. We implemented a software timer on the micro:bit: on the first button press, a short countdown is started. If a second press occurs within the interval, it is treated as a double press; otherwise it is treated as a single press. This reliably detects double-press events in `microbit.py`.

- **Synchronization Delay:** Browser actions can sometimes take time to update the page state (e.g. volume changes) after a click. We added a short `sleep()` delay in `control_saavan.py` immediately after sending each command. This gives the webpage time to update before the script proceeds, ensuring that subsequent checks are reliable.

- **Serial Noise/Corruption:** On occasion, the serial link may deliver spurious or malformed data. To handle this, `usb_controller.py` wraps the serial `read()` and decoding in a try/except block. Any invalid bytes are ignored, so only properly decoded command characters trigger actions. This simple error handling prevents random noise from causing failures.

Results

The completed system **successfully implements all basic music controls on the JioSaavn player.** During testing, each control command (play/pause, next, previous, volume up/down) worked reliably on the web interface. The system is responsive: the typical latency from pressing a button (or performing a gesture) to the action occurring on JioSaavn is on the order of 0.5–1.0 seconds. The table below summarizes representative latency measurements for each action:

Action	Latency (seconds)
Play/Pause	0.6
Next Track	0.7
Previous Track	0.7
Volume Up	0.8
Volume Down	0.8

Future Improvements

- **Implement a wireless Bluetooth-enabled controller:** Currently, the Micro:bit is connected via a wired setup. In the future, using Bluetooth would make the controller truly wireless, improving user convenience and allowing more accurate and responsive gesture detection (like tilting, shaking) without physical constraints.

- Implement **gesture-based volume control**, e.g. tilting the micro:bit forward/backward to continuously adjust volume.

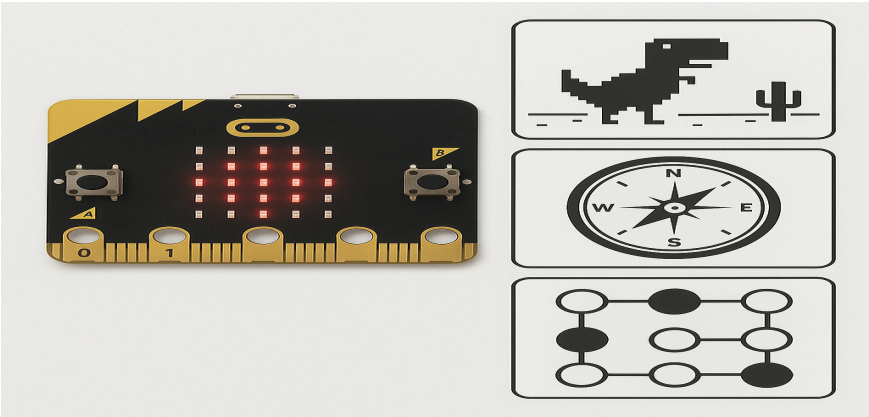
- Provide **visual feedback on the micro:bit** itself, such as showing a volume bar on the LED matrix when the volume is being changed in real time or scrolling the current track name.

- Add a **Mute/Unmute** command and indicator (currently volume control only adjusts level).'

Conclusion

This project met its objectives by enabling a **micro:bit v2 to control the JioSaavn web player through a USB serial link**. All key functions **(play/pause, track navigation and volume control)** were implemented and tested to work reliably as shown in the video uploaded, with each action executed in under about one second. The results demonstrate that a simple MicroPython program on a micro:bit can interface with a Python/Selenium program on a PC to manipulate a modern web application. This validates the feasibility of bridging low-cost microcontroller hardware with online services for intuitive, physical control of web-based media.

PTO: Additional Fun Stuff :)



Side Hustles: Exploring More with Micro:bit

After successfully building the JioSaavn Music Controller, our curiosity and enjoyment with the BBC micro:bit platform led us to explore further. What began as a course assignment quickly turned into a playground of creativity and experimentation. We ended up building **three additional mini-projects**, purely out of interest and excitement. These side hustles were **wireless (not connected to a PC)**, self-contained, and tested the diverse capabilities of the micro:bit.

Video Demonstrations of Side Hustle Projects

We've recorded brief demo videos showcasing each of our mini-projects. You can view them here:

- **Chrome Dino Game:**

https://drive.google.com/file/d/1vZhc-wmto0BQOWm81tPAqW0lz1AOGBVR/view?usp=share_link

- **Digital Compass:**

https://drive.google.com/file/d/13PJoXrYQ3vEFnqC_ez0mqVcEbzZqfWS9/view?usp=share_link

- **Memory Game:**

https://drive.google.com/file/d/1PsbJtFaRprxVQib4Qu50ni7nIJhfd5Ud/view?usp=share_link

- ❖ Separate Reports have been attached for these Side Hustles for reference.
- ❖ If the Google Drive links do not work, try copy pasting them to the web browser and remove any additional white spaces in the link copied.