Puppet Show — Firmware Development Log

Raspberry Pi Pico W · EarlePhilhower Arduino core · PlatformIO

FabAcademy · Mechanical scroll puppet show with MP3 sync · 2026-07-04

EXCHANGE 1

Initial hardware description and project setup

✓ Reconstructed — original conversation lost

ASSUMED

Building a mechanical puppet show. Paper scrolls with drawn scenes advance automatically in sync with MP3 audio. Set up a PlatformIO project and write the firmware for this hardware:

Component	Pins
Button 1 — backward / rewind	GP14
Button 2 — forward / teach	GP15
SM1 (supply reel) STEP / DIR / EN	GP28 / GP27 / GP26
SM2 (take-up reel) STEP / DIR / EN	GP19 / GP18 / GP22
XY-V17B MP3 TX / RX	GP12 / GP11
MP3 BUSY	GP13
Slot sensor	GP10
MFRC522 RFID SS / RST	GP5 / GP0
NeoPixel LED	GP16

SM2 winds the take-up reel to advance the scroll; SM1 rewinds it. Only one motor should be active at a time. RFID card tap should jump to a scene. Each scene has a matching MP3 track on the SD card.

RESPONSE

PlatformIO project created. Pin definitions in `include/config.h`. Full state machine in `src/main.cpp`.

- State machine: IDLE → AUDIO_PLAYING → ADVANCING → REWINDING → FINISHED
- Motor mutual exclusion enforced in `enableMotor1()` / `enableMotor2()`
- XY-V17B driven natively — 0xAA frame protocol, 9600 baud
- NeoPixel shows state: green=idle, cyan=playing, yellow=advancing, red=error
- RFID UID byte maps to scene number and jumps directly

EXCHANGE 2

Individual component test programs

✓ Reconstructed — original conversation lost

ASSUMED Create separate test programs for each component — steppers, MP3 module, and buttons — so I can verify them individually before running the full firmware.

RESPONSE Three test projects created under `test/`, each with its own `platformio.ini`:

- `stepper_test/` — runs SM1 and SM2 separately, 3 revolutions each, prints direction and step count
- `mp3_test/` — plays track 1, monitors BUSY pin, replays on finish, heartbeat every 3 s
- `button_test/` — button jog with position tracking (expanded in later exchanges)

EXCHANGE 3

Stepper speed too fast — 30 RPM calibration

PROMPT The motor is pulsing too fast. Using DRV8825 at 1/32 microstepping with NEMA17 steppers. Run at 30 RPM.

RESPONSE Recalculated step delay: 6400 steps/rev × 0.5 rev/s (30 RPM) = 3200 steps/s → **156 µs per step**. Updated `STEP_DELAY_US` from 800 → 156.

Physical test confirmed motor directions: SM2 DIR=HIGH winds forward; SM1 DIR=HIGH rewinds.

EXCHANGE 4

MP3 serial crashes — fixed with SerialPIO

PROMPT MP3 serial is not working. Do not change the pins — they are already on the PCB.

RESPONSE `Serial2.setRX(11)` crashes because GP11 is not a hardware UART pin on RP2040. Pins kept the same. Switched to `SerialPIO` — PIO-based software UART that works on any GPIO pair.

```
// Before — crashes at boot:
Serial2.setRX(11); Serial2.setTX(12); Serial2.begin(9600);

// After — working:
#include <SerialPIO.h>
SerialPIO mp3PIO(12, 11, 32);   // TX=GP12  RX=GP11
#define MP3_SERIAL mp3PIO
```

EXCHANGE 5

BUSY pin inverted + SD card file naming

PROMPT Playing in loop but no sound. Only one file on the SD card: `001.mp3`.

RESPONSE Two bugs found and fixed:

- BUSY pin coded as LOW=playing. Datasheet says HIGH=playing. Fixed to `digitalRead(MP3_BUSY_PIN) == HIGH`
- SD file must be 5 digits. Renamed `001.mp3` → `00001.mp3`

Audio still silent after fixes. Likely cause: CON1/CON2/CON3 jumpers on PCB not configured for UART mode (need CON3=HIGH, CON2=LOW, CON1=LOW). Also check DACL/DACR wired to amplifier input, not SPK pins. Left for a later session.

EXCHANGE 6

Uploading without BOOTSEL — picotool

PROMPT Can we upload without pressing BOOTSEL every time? It's frustrating.

RESPONSE Switched to `upload_protocol = picotool` in all test `platformio.ini` files. Picotool resets the Pico into bootloader over USB — no button needed. Requires Zadig WinUSB driver installed once.

```
upload_protocol = picotool
monitor_speed   = 9600
monitor_port    = COM4    # check Device Manager — may be COM3
```

Also replaced `while(!Serial)` with `delay(3000)` so the board doesn't hang when no monitor is open.

EXCHANGE 7

Button jog — forward and backward scroll control

PROMPT Button 2 → scroll forward. Button 1 → scroll backward.

RESPONSE Jog control added to `button_test/`. Pressing Button2 enables SM2 (forward). Pressing Button1 enables SM1 (backward). Motor stops on release. Global `position` counter tracks steps from origin and prints every 50 steps with both steps and revolutions.

EXCHANGE 8

Auto sequence — 3 s hold to run, emergency stop

PROMPT Hold both buttons for 3 seconds → advance scroll, wait 5 s, repeat 4 more times. Any button press during the run = emergency stop.

RESPONSE Three debounce values added to prevent false triggers:

- `BOTH_HOLD_MS = 3000` — minimum hold before auto starts
- `BOTH_RELEASE_DEBOUNCE = 80 ms` — brief bounces during the hold don't reset the timer
- `ESTOP_DEBOUNCE_MS = 500 ms` — ignores buttons for 500 ms after start so releasing the hold doesn't immediately stop the sequence

E-stop checked every 100 steps during advance and every 10 ms during the 5 s wait.

EXCHANGE 9

Teach and replay — 5 custom scene lengths

PROMPT Instead of fixed rotations, I want to jog to each scene boundary and save those distances. Need 5 unequal scenes — just these 2 buttons.

RESPONSE Teach-and-replay built entirely around the 2 existing buttons:

- Jog forward with Button2 → release saves that leg's distance
- Repeat 5 times, one per scene boundary
- Rewind to start with Button1
- Hold both 3 s → plays all 5 saved distances with 5 s pauses

```
long sceneDist[5];       // filled during teach mode
bool teachDone = false;  // gates the auto sequence
```

EXCHANGE 10

Scene distance — confirmed 6.17 rotations

PROMPT Print current position as it moves. One scene is about 650 steps — update it. It's exactly 6.17 rotations.

RESPONSE Position prints every 50 steps showing both steps and revolutions. Scene distance converged through testing: 650 → 28,800 (4.5 rev) → **39,488 steps (6.17 rev × 6400)**.

Pre-filled into all 5 scene slots so auto play works immediately without a teach session:

```
long sceneDist[5] = {39488, 39488, 39488, 39488, 39488};
bool teachDone = true;
```

Unit: 1 position unit = 1 microstep = 1/6400 revolution (at 1/32 microstepping).

EXCHANGE 11

Session log exported

PROMPT Convert this conversation into a downloadable log file.

RESPONSE Markdown log written to `Puppet Show/SESSION_LOG_2026-07-04.md`. Covers hardware pinout, bugs and fixes, motor directions, scene calibration, test program summaries, pending items, and next steps.

BUILD STATUS

✓ Stepper jog	✓ Teach and replay
✓ Emergency stop	✓ BUSY pin detection
○ MP3 audio output	○ Slot sensor + full show run