PaperMono Microphone

APIs and example programs for the PaperMono microphone.

Example Programs

Build Requirements

- M5Stack Board Manager version >= 3.3.9
- Board option = `M5PaperMono`
- M5Unified library version >= 0.2.20
- M5GFX library version >= 0.2.27
- M5IOE1 library version >= 1.0.9

cpp

```cpp
#include <Arduino.h>
#include <M5IOE1.h>
#include <M5Unified.h>
#include <driver/gpio.h>
#include <math.h>

namespace {

constexpr uint32_t kSampleRateHz = 16000;
constexpr size_t kSampleCount = 1600;
constexpr uint32_t kUpdateIntervalMs = 1000;
constexpr size_t kHistoryLength = 60;

constexpr int kPdmClockPin = 45;
constexpr int kPdmDataPin = 46;
constexpr auto kPdmPowerPin = M5IOE1_PIN_12;

constexpr int kGraphLeft = 42;
constexpr int kGraphTop = 245;
constexpr int kGraphWidth = 396;
constexpr int kGraphHeight = 430;

M5IOE1 ioe1;
M5Canvas canvas(&M5.Display);
int16_t samples[kSampleCount];
float levelHistory[kHistoryLength] = {};
size_t historyCount = 0;
uint32_t nextUpdateMs = 0;

struct VolumeReading {
    float rms;
    float peak;
    float dbfs;
    float percent;
};

[[noreturn]] void stopWithError(const char* message)
{
    Serial.println(message);
    canvas.fillSprite(TFT_WHITE);
    canvas.setTextColor(TFT_BLACK, TFT_WHITE);
    canvas.setTextDatum(middle_center);
    canvas.setFont(&fonts::FreeSansBold18pt7b);
    canvas.drawString("MIC ERROR", 240, 330);
    canvas.setFont(&fonts::FreeSans12pt7b);
    canvas.drawString(message, 240, 410);
    M5.Display.setEpdMode(epd_mode_t::epd_fastest);
    canvas.pushSprite(0, 0);
    while (true) {
        delay(1000);
    }
}

bool waitForRecording(uint32_t timeoutMs)
{
```

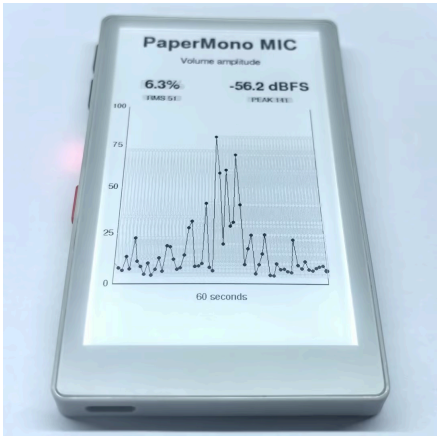
```
    const uint32_t startedAt = millis();
    while (M5.Mic.isRecording()) {
        if (millis() - startedAt >= timeoutMs) {
            return false;
        }
        delay(1);
    }
    return true;
}

VolumeReading analyzeSamples()
{
    int64_t sum = 0;
    for (size_t i = 0; i < kSampleCount; ++i) {
        sum += samples[i];
    }
    const double mean = static_cast<double>(sum) / kSampleCount;

    double squareSum = 0.0;
    double peak = 0.0;
    for (size_t i = 0; i < kSampleCount; ++i) {
        const double centered = static_cast<double>(samples[i]) - mean;
        squareSum += centered * centered;
        peak = max(peak, fabs(centered));
    }

    const float rms = sqrt(squareSum / kSampleCount);
    const float dbfs = rms > 0.0f ? 20.0f * log10f(rms / 32768.0f) : -96.0f;
    const float percent = constrain((dbfs + 60.0f) * (100.0f / 60.0f), 0.0f, 100.0f);
    return {rms, static_cast<float>(peak), dbfs, percent};
}

void appendHistory(float value)
{
    if (historyCount < kHistoryLength) {
        levelHistory[historyCount++] = value;
        return;
    }

    memmove(levelHistory, levelHistory + 1,
            (kHistoryLength - 1) * sizeof(levelHistory[0]));
    levelHistory[kHistoryLength - 1] = value;
}

void drawGrid()
{
    canvas.drawRect(kGraphLeft, kGraphTop, kGraphWidth, kGraphHeight, TFT_BLACK);
    canvas.setFont(&fonts::FreeSans9pt7b);
    canvas.setTextDatum(middle_right);

    for (int percent = 0; percent <= 100; percent += 25) {
        const int y = kGraphTop + kGraphHeight -
                      (percent * kGraphHeight / 100);
        const uint16_t color = percent == 0 ? TFT_BLACK : TFT_LIGHTGREY;
        canvas.drawFastHLine(kGraphLeft, y, kGraphWidth, color);
```

```
        canvas.drawString(String(percent), kGraphLeft - 7, y);
    }

    canvas.setTextDatum(top_center);
    canvas.drawString("60 seconds", kGraphLeft + kGraphWidth / 2,
                      kGraphTop + kGraphHeight + 18);
}

void drawScreen(const VolumeReading& reading)
{
    canvas.fillSprite(TFT_WHITE);
    canvas.setTextColor(TFT_BLACK, TFT_WHITE);
    canvas.setTextDatum(top_center);

    canvas.setFont(&fonts::FreeSansBold24pt7b);
    canvas.drawString("PaperMono MIC", 240, 24);
    canvas.setFont(&fonts::FreeSans12pt7b);
    canvas.drawString("Volume amplitude", 240, 92);

    canvas.setFont(&fonts::FreeSansBold18pt7b);
    canvas.setTextDatum(middle_center);
    canvas.drawString(String(reading.percent, 1) + "%", 110, 180);
    canvas.drawString(String(reading.dbfs, 1) + " dBFS", 350, 180);

    canvas.setFont(&fonts::FreeSans9pt7b);
    canvas.drawString("RMS " + String(reading.rms, 0), 110, 220);
    canvas.drawString("PEAK " + String(reading.peak, 0), 350, 220);

    drawGrid();

    if (historyCount > 0) {
        const int plotLeft = kGraphLeft + 1;
        const int plotRight = kGraphLeft + kGraphWidth - 2;
        const int plotBottom = kGraphTop + kGraphHeight - 1;
        const int plotHeight = kGraphHeight - 2;
        int previousX = plotRight -
                        static_cast<int>((historyCount - 1) *
                                         (plotRight - plotLeft) /
                                         (kHistoryLength - 1));
        int previousY = plotBottom -
                        static_cast<int>(levelHistory[0] * plotHeight / 100.0f);

        for (size_t i = 1; i < historyCount; ++i) {
            const int x = plotRight -
                          static_cast<int>((historyCount - 1 - i) *
                                           (plotRight - plotLeft) /
                                           (kHistoryLength - 1));
            const int y = plotBottom -
                          static_cast<int>(levelHistory[i] * plotHeight / 100.0f);
            canvas.drawLine(previousX, previousY, x, y, TFT_BLACK);
            canvas.fillCircle(x, y, 3, TFT_BLACK);
            previousX = x;
            previousY = y;
        }
        canvas.fillCircle(previousX, previousY, 4, TFT_BLACK);
```

```
    }

    M5.Display.setEpdMode(epd_mode_t::epd_fastest);
    canvas.pushSprite(0, 0);
}

void initializeMicrophone()
{
    const m5ioe1_err_t ioeError = ioe1.begin(
        &M5.In_I2C, M5IOE1_DEFAULT_ADDR_2, M5IOE1_I2C_FREQ_100K);
    if (ioeError != M5IOE1_OK) {
        stopWithError("M5IOE1 initialization failed");
    }

    ioe1.pinMode(kPdmPowerPin, OUTPUT);
    if (ioe1.setDriveMode(kPdmPowerPin, M5IOE1_DRIVE_PUSHPULL) != M5IOE1_OK) {
        stopWithError("PDM power pin setup failed");
    }
    m5ioe1_err_t powerError = M5IOE1_OK;
    ioe1.digitalWriteWithRes(kPdmPowerPin, HIGH, &powerError);
    if (powerError != M5IOE1_OK) {
        stopWithError("PDM microphone power failed");
    }
    delay(20);

    gpio_reset_pin(static_cast<gpio_num_t>(kPdmDataPin));
    gpio_set_direction(static_cast<gpio_num_t>(kPdmDataPin), GPIO_MODE_INPUT);
    gpio_set_pull_mode(static_cast<gpio_num_t>(kPdmDataPin), GPIO_FLOATING);
    gpio_reset_pin(static_cast<gpio_num_t>(kPdmClockPin));

    M5.Speaker.end();
    M5.Mic.end();
    auto micConfig = M5.Mic.config();
    micConfig.pin_mck = I2S_PIN_NO_CHANGE;
    micConfig.pin_bck = I2S_PIN_NO_CHANGE;
    micConfig.pin_ws = kPdmClockPin;
    micConfig.pin_data_in = kPdmDataPin;
    micConfig.i2s_port = I2S_NUM_0;
    micConfig.input_channel = m5::input_channel_t::input_only_right;
    micConfig.sample_rate = kSampleRateHz;
    micConfig.over_sampling = 1;
    micConfig.magnification = 2;
    micConfig.dma_buf_len = 128;
    micConfig.dma_buf_count = 8;
    M5.Mic.config(micConfig);

    if (!M5.Mic.begin()) {
        stopWithError("PDM microphone begin failed");
    }
    Serial.println("PDM microphone: ready");
}

}  // namespace

void setup()
```

```
{
    Serial.begin(115200);
    delay(100);
    Serial.println("PaperMono microphone amplitude demo");

    auto config = M5.config();
    config.clear_display = false;
    config.internal_rtc = false;
    config.internal_imu = false;
    config.internal_mic = false;
    M5.begin(config);

    M5.Display.setRotation(0);
    M5.Display.setBrightness(160);
    M5.Display.setEpdMode(epd_mode_t::epd_fastest);
    canvas.setColorDepth(16);
    if (!canvas.createSprite(M5.Display.width(), M5.Display.height())) {
        while (true) {
            Serial.println("Display canvas allocation failed");
            delay(1000);
        }
    }

    initializeMicrophone();
    nextUpdateMs = millis();
}

void loop()
{
    M5.update();
    const uint32_t now = millis();
    if (static_cast<int32_t>(now - nextUpdateMs) < 0) {
        delay(5);
        return;
    }
    nextUpdateMs = now + kUpdateIntervalMs;

    if (!M5.Mic.record(samples, kSampleCount, kSampleRateHz, false) ||
        !waitForRecording(250)) {
        Serial.println("Microphone sample failed");
        return;
    }

    const VolumeReading reading = analyzeSamples();
    appendHistory(reading.percent);
    Serial.printf("MIC rms=%.1f peak=%.1f dbfs=%.1f level=%.1f%%\n",
                  reading.rms, reading.peak, reading.dbfs, reading.percent);
    drawScreen(reading);
}
```

The program uses the M5IOE1 to enable power to the PDM microphone. It captures 1,600 samples per second at a 16 kHz sampling rate, removes the DC offset, and calculates the RMS, peak, dBFS, and volume percentage. The screen shows the current values and a volume graph covering the most recent 60 seconds, while the data is also output to the Serial Monitor.



API

The PaperMono microphone features use `Mic_Class` from the `M5Unified` library. For more information, refer to the following documentation:

- M5Unified - Mic Class