Prac 4 - SPI and Threading

Matthew Terblanche† and Michael Wetzel‡
EEE3096S Class of 2019
University of Cape Town
South Africa
†TRBMAT002 ‡WTZMIC001

Introduction

The Raspberry Pi's many peripheral ports is one of it's most attractive features. One of them being an audio jack. The Raspberry Pi uses pulse width modulation to play its audio through the jack and is unfortunately not the best way to play audio as it requires a lot of bandwidth and there is an alternation in power used.

In this practical the use of a DAC (Digital to analogue converter) to play audio through the audio jack is investigated. The DAC used is the MCP4911 which is a 10-bit device. The audio to be played is 8bit at 16kHz, which is very low for standard audio quality. This practical is however not aimed at producing good sound but rather at understanding and getting practise with SPI.

Playing audio has a hard time requirement, which means it has certain things that it must play at specific times. If this does not happen, sound is distorted. It is therefore key that the audio be ready to play at the right times. To ensure this, a similar technique to circular buffering is used which consists out of one 3D array. Data from a text file will be read into the "Read" column of the list and only when this is done will the program try to play it. This will prevent errors where the program is trying to play audio that is not yet loaded.

A 8 bit character will be read from a text file with 4 control bits in the front appended into the list and then sent to the DAC via Wiring Pi's modules. The DAC will then convert these digital signals into analog signals and play it.

The speed of the audio played can be set by adjusting the SPI clock speed of the Pi.

SPI communication using Wiring Pi:

(a) Initialisation

To setup the SPI channel the function " wiringPiSPISetup(SPI_CHAN, SPI_SPEED); " must be called [1]. With SPI_CHAN defined as 0 for channel 0 and SPI_SPEED defined as 409 600 Hz. The SPI_SPEED or clock speed is calculated using $SPICLOCK = SAMPLE\ RATE \times WIDTH \times No.CHANNELS \times \frac{8}{5}$ with the SAMPLE RATE being 16 000, the WIDTH 16, and No.CHANNELS 1.

(b) Send data

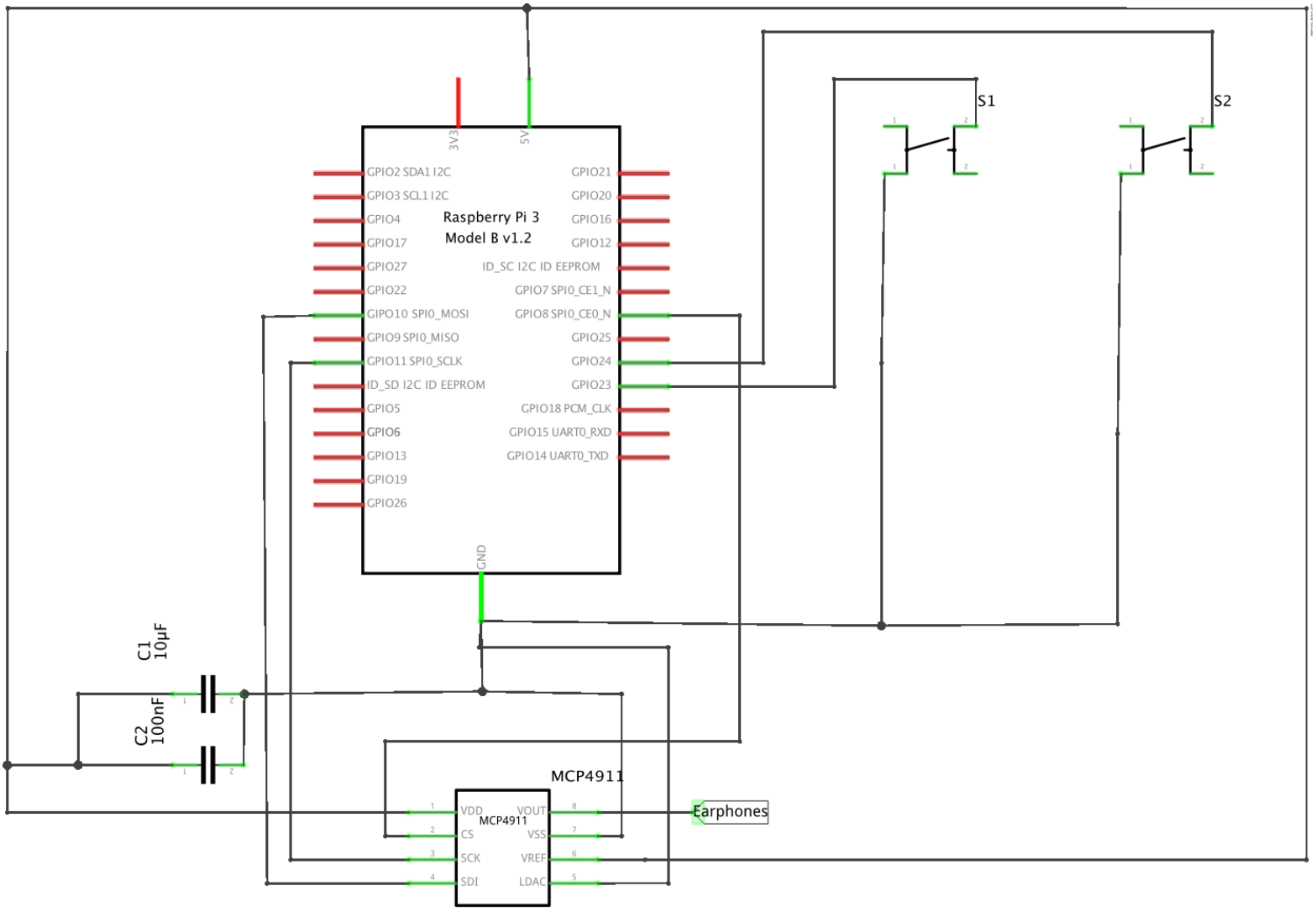
To send data to the DAC the data from the raw audio file was read into a buffer using the function " buffer[bufferWriting][counter][0] = 0b01110000 | (ch >> 4); ". This added the 4 config bits and the remainder data bits with a bit shift of 4 to the right. The function " buffer[bufferWriting][counter][1] = ch << 4; " then added the remainder data bits to the buffer. The config bits are defined so that the write command is set to write to the DAC register, the V_{REF} input buffer is buffered, the output gain is 1x, and the DAC is set to active mode operation. The data in the buffer is then sent to the DAC using " wiringPiSPIDataRW(SPI_CHAN, buffer[bufferReading][buffer_location], 2); ". This sends all 2 bytes or 16 bits of data in the buffer to the DAC which is then converted to an analog signal which is played through earphones or an amplifier.

The importance of real time constraints, and why the Raspberry Pi (under Raspbian) is unable to implement these

Real time constraints are very important when doing time related tasks and when certain events have to happen within a certain time period. The most important real time constraints are hard real time constraints. When a hard real time constraints is not implemented properly it is a matter of life and death. Hard real time constraints are used in medical equipment for example [2].

Raspbian is a linux based operating system and is unable to implement real time constraints because it is a general purpose operating system and not a real time operating system. A general purpose operating system works on more of a task based schedule and is not time critical. Where a real time operating system is time based and is time critical [3].

Circuit Diagram



References

[1] “WiringPi Documentation”
http://wiringpi.com/reference/spi-library/

[2] "Arduino vs. Raspberry Pi: 9 Crucial Differences You Need to Know"
https://appcodelabs.com/arduino-vs-raspberry-pi-9-crucial-differences

[3] “Differences between a GPOS (Normal OS) and an RTOS (Real Time OS)”
http://www.circuitstoday.com/gpos-versus-rtos-for-an-embedded-system