Electronics A2 Project

Microcontroller-based Oscilloscope

Author:
Harry Rickards

2014

Contents

List of Figures

Chapter 1

Problem Analysis and Solution Design

1.1 Problem Definition

CROs and DSOs

Oscilloscopes available in secondary schools or sixth forms are generally basic analogue oscilloscopes known as CROs, or Cathode Ray Osciloooscopes (named so because the display is a Cathode Ray Tube).[1]

The alternative is a DSO, or Digital Storage Oscilloscope. These work in a very different way to CROs. Like in a CRO, the input signal is first passed through some analogue circuitry to amplify it, and so on. However, it's then sampled into a digital signal using a DAC (Digital to Analogue Converter) and processed using a microcontroller (perhaps using the help of an FPGA[2]or a DSP[3]), before being output onto some sort of digital screen.[4]

Advantages of DSOs

DSOs offer a number of advantages over CROs. As well as relatively basic differences such as the traces being much more sharply defined, DSOs allow digital signal processing to be done on the input signal. This means that, for example, DFTs (Discrete Fourier Transforms)[5] can be run to produce a frequency spectrum of the input signal, allowing the

[1]Phillips, *Using an Oscilloscope.*

[2]**Field Programmable Gate Array** a device that can be programmed to ~become a very large logic circuit

[3]**Digital Signal Processor** a custom microcontroller especially designed to perform various *signal processing* functions (e.g., performing a Discrete Fourier Transform to obtain the frequency spectrum of a signal, or using Principal Component Analysis to seperate two audio signals)

[4]TiePie Engineering, *Digital Data Acquisition.*

[5]While DFTs are the ones most famous transformations, mainly due to the prevalence of the FFT (Fast Fourier Transform) algorithm, in practice DCTs (Discrete Cosine Transforms) are more commonly used as they can be calculated quicker to achieve a similar end effect

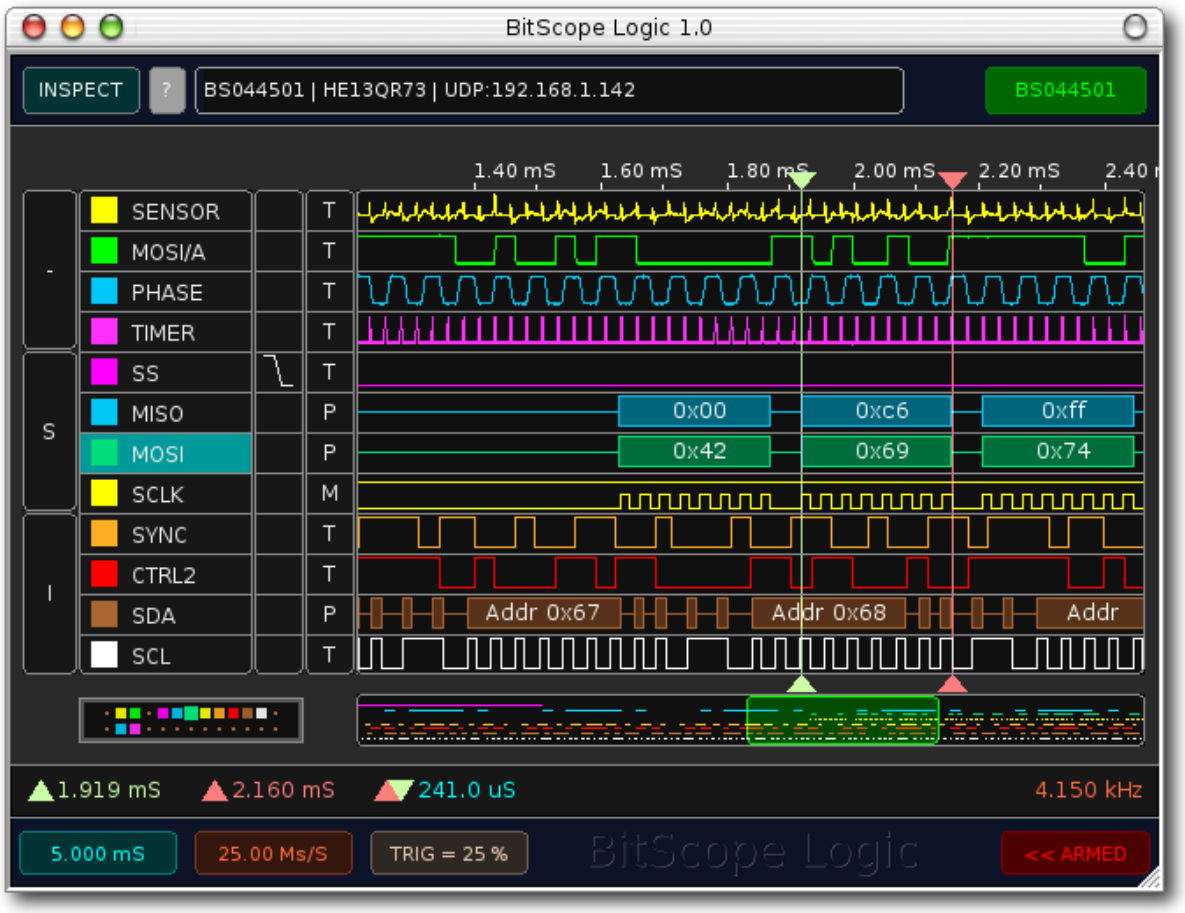

Figure 1.1: Example output from a Logic Analyser[9]

user to see both time-domain and frequency-domain representations of the signal.[6]

Additionally, a DSO can be used to create a basic Logic Analyser. This is essentially the digital equivalent to an oscilloscope — a relatively large number of digital wires and buses are connected to the Logic Analyser and a timing diagram is output[7].[8]

Problem Definition

This project aims to create a Digital Sampling Oscilloscope, with the following more advanced features included:

- Frequency spectrums of the input signals available
- Basic capability as a Logic Analyser

From now on, oscilloscope will be used to refer to a Digital Sampling Oscilloscope and CRO will be used to explicitly refer to a Cathode Ray Oscilloscope.

[6]This also means that DSOs can be used as basic spectrum analysers: essentially devices that plot frequency spectrums, these have uses as far ranging as determining GSM interference when planning mobile phone networks

[7]In more advanced, standalone Logic Analysers, more complicated outputs such as decoded Ethernet packets can be output

[8]Woodward, *What's the Difference Betwen a Mixed-Signal Oscilloscope And A Logic Analyzer?*

1.2 Research into Existing Oscilloscopes

Using two articles from Tong[10] and Gabotronics,[11] digital oscilloscopes can be categorised by the following criteria:

- **Bandwidth** The bandwidth[12] in Hz of the initial analogue stages of the oscilloscope. Common values range from 100 kHz to 100 MHz.
- **Type of Sampling** There are two common types of sampling, real-time and equivalent-time sampling. With real-time sampling the scope samples sequentially over the wave, whereas with equivalent-time sampling the scope captures a sample from a different part of the wave every time period. This is perhaps illustrated more clearly in section 1.2.

  While equivalent-time sampling offers a much faster sampling rate, it is more complicated to implement so for the purposes of this project we will only be considering real-time sampling.
- **Sample Rate** The rate (in Hz or the equivalent *samples per second*) at which the oscilloscope can take individual digital samples from the signal. Common values range from 10 kHz to 1 GHz.
- **Resolution** The number of bits each sample has. Common values are 8, 12 and 16 bits. This determines the precision of the oscilloscope, and hence the minimum noise.
- **Memory Depth** The number of samples the oscilloscope can store at one time. Measured in either words, or bits (for example, if the resolution is 8 bits then a depth of 128 words is equivalent to 1024 bits). Intuitively, this can be thought of as the horizontal resolution. Common values range from 100 samples to 10 M samples.
- **Triggers Available** The different options to trigger the oscilloscope to start capturing samples, with the most common one being edge triggering[13] (when the signal passes a threshold on a rising edge[14]). There are also more advanced triggers such as pulse width triggering, which triggers the scope when a digital pulse of a certain width is detected. Some expensive oscilloscopes can provide extremely complicated triggers: e.g., decoding Ethernet, TCP/IP[15] and HTTP[16] packets from a digial input, and eventually triggering the analogue inputs when a certain webpage is requested.
- **Input range** The minimum and maximum range of input voltages that can be detected by the oscilloscope. A typical value would be from ±50 mV to ±50 V. In many cases, this will be a relatively small range, but an active scope probe (a test probe with an amplifier inside it) will be used to broaden this range.

[10] Tong, *What to Look for When Choosing an Oscilloscope.*

[11] Gabotronics, *Digital oscilloscopes for hobbyists.*

[12] Most commonly, bandwidth is taken to be the range of fraquencies where the signal has an amplitude gain of $-3\,\text{dB} \approx 71\%$

[13] Picotech, *Advanced Triggering with PicoScope.*

[14] A falling edge could also be used, but the type of edge is part of the trigger specifications and stays constant

[15] The protocol used for internet networking

[16] The protocol used to transfer web pages over the internet

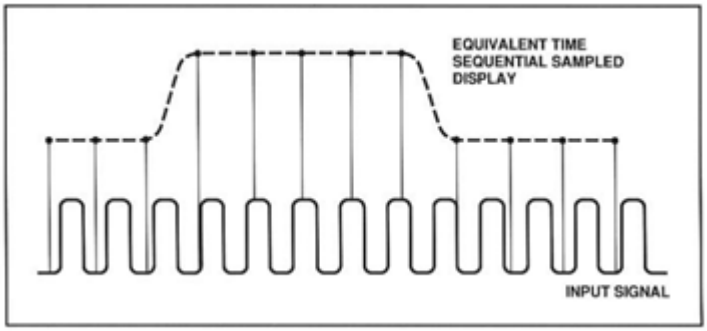

Figure 1.2: Diagram illustrating equivalent time sampling[17]

1.3 Practical Investigations

1.3.1 Breadboard Frequency Investigation

A limiting factor in the bandwidth and sampling rate[18] will be due to stray capacitances and inductances on the breadboards[19] used to construct the system, which will cause a great deal of noise at high frequencies.

Because of this, the first practical investigation undertaken was to find the attenuation of a signal running through a breadboard at various frequencies.

A signal generator was connected to an oscilloscope through a number of connections on a breadboard, and the amplitude of the final signal recorded at various frequencies. The same experiment was then repeated connecting the oscilloscope directly to the signal generator.

For each frequency, the gain in dB was then calculated using the equation

$$\begin{aligned}
\text{Gain} &= 10\log_{10}\left(\frac{P_{out}}{P_{in}}\right) \\
&= 10\log_{10}\left[\frac{\frac{(V_{out})^2}{R_{out}}}{\frac{(V_{in})^2}{R_{in}}}\right] \\
&\approx 10\log_{10}\left[\frac{(V_{out})^2}{(V_{in})^2}\right] \\
&= 20\log_{10}\left(\frac{V_{out}}{V_{in}}\right)
\end{aligned}$$

Photos and results

[18]The sampling rate will be influenced by this much more than one would expect — for example, with the MAX114 ADC, to capture samples at 1 MHz the ADC must be communicated with at 15 MHz (Maxim Integrated, *MAX114/MAX118 Datasheet*)

[19]As detailed later on, a PCB will be used for the final system, however prototyping will occur on breadboards so the maximum frequency on a breadboard is still a limiting factor

1.3.2 Oscilloscope Reliability Investigation

An existing oscilloscope (in this case, a CRO) will be a vital part of testing the osilloscope, so it would be wise to test it's accuracy first.

An FPGA[20]was programmed to produce a square wave at a number of different frequencies, and the wave was looked at on an oscilloscope. Due to their completely different nature compared to microcontrollers, FPGAs are usually programmed in a language called Verilog. While the details are beyond the scope of this project[21], let us just say that Verilog is a way of describing the propogation of signals based on dependencies on both time and other signals.[22]

The FPGA was programmed to perform a 50% duty-cycle clock divisions of the FPGA's internal[23] 50 MHz clock, meaning the output frequency could be easily calculated using

$$f = \frac{50\,\text{MHz}}{2^n}$$

where n is the clock divisor.

Photos and results Verilog code

1.4 Numerical Parameters

Frequency and Sampling Rate

The sampling rate should be as high as feasibly possible to allow the oscilloscope a wide variety of uses. By Shannon ("A Mathematical Theory of Communication"), the maximum frequency[24] that we can sample is half the sampling rate. In reality, because of the basic real-time sampling method being used it must be at the very least 25 times less than the sampling rate (allowing just over 12 samples for each half-period of the wave).

ADCs in a DIP[25] package (suitable for breadboards) are not readily available beyond 2 MHz. Furthermore, the only 2 MHz DIP ADC, the AD7822, is renowned for being par-

[20]**Field Programmable Gate Array** a device that can be programmed to ~become a very large logic circuit

[21]Tala (*ASIC World*), found online at `http://www.asic-world.com/verilog/index.html`, is an excellent resource for learning Verilog

[22]Wikipedia, *Verilog — Wikipedia, The Free Encyclopedia.*

[23]A high-quality crystal oscillator

[24]Actually, that's the maximum sinusoidal frequency that we can sample. By Fourier, all other waveforms can be represented as a sum of sinusoids, and in reality sampling at 25 times the frequency of the waveform means that we'll be able to get a high enough amount of detail for most non-sinusoidal waveforms

[25]Electronics components come in a variety of different packages of many different shapes and sizes. Generally, those that fit in a breadboard are part of the DIP (dual in-line package) family: e.g., CDIP (Ceramic DIP) and PDIP (Plastic DIP). Other packages, such as TSOP, are usually much smaller and suitable only for soldering onto PCBs. While a PCB will be used for the final oscilloscope, soldering surface-mount chips (a generic term for all packages smaller than DIP, such as TSOP) requires unavailable specialist equipment.

ticularly difficult to interface with a microcontroller. Instead, an ADC with a maximum sampling rate of 1 MHz will be used (many DIP ICs can provide this sampling rate).

However, at such a high speed getting the full sampling rate out of an ADC is not usually possible with a microcontroller (because, for example, a control bit might have to go high then low again for 20 ns, but a microcontroller with a 16 MHz clock can only do this in an absolute minimum of 62.5 ns). So the oscilloscope sampling rate will be specified as at least 500 kHz.

This means the maximum frequency that should be sampled is $\frac{500\,\text{kHz}}{25} = 20\,\text{kHz}$.

Bandwidth

The maximum frequency into the analogue amplification stage will be 20 kHz, but we also need to take into consideration harmonics (remember, by Fourier any non-sinusoidal signal can be represented as a sum of sinusoidal signals of increasing frequency). To be safe, we will specify a minimum bandwidth of 1 MHz.

Resolution

Inaccuracies in the oscilliscope output are more likely to be caused by the analogue circuitry (in particular, the digital potentiometers are only accurate to 0.25%) than a low resolution, so an 8 bit resolution will suffice (this offers an accuracy of $\approx 0.4\%$, more accurate than the digital potentiometer accuracy). This also means ADCs will be more readily available, as 8 bit DIP ADCs are more common than higher resolution ones.

Memory Depth

The memory depth needs to be large enough to have a horizontally precise signal, but small enough that the samples can easily be stored in the microcontroller. For these reasons, 1024 words (equivalent to 8192 bits or 1 kB with an 8 bit resolution) will be chosen. Most microcontrollers have multiple kB of RAM, so multiple signals can easily be stored in the RAM.

Input Range

For the minimum input range, we'll choose the standard ±50 mV, however high-bandwidth rail-to-rail op amps are only cheaply available up to ±5 V so for the maximum input range we'll choose ±5 V[26].

[26]While the input to the op amp could be a higher voltage than the supply voltage, a digital potentiometer will be needed to automatically adjust the gain of the op amp. These only work up to to the supply voltage.

Triggers Available

For the purposes of this project, simple edge triggering will more than suffice. Both the channel on which and the threshold at which the trigger occurs will be adjustable by the user.

1.5 Communication with Android Device

As detailed in section 2.1.6, an Android device will be used as the frontend to the oscilloscope. This means that the microcontroller must somehow communicate with the device.

There are 4 feasible ways that this could be achieved: via WiFi, Bluetooth and USB.

USB

This would perhaps be the easiest way to implement the communication. Small discrete devices, such as the CP2102,[27] convert between the relatively complicated USB protocol and the simple UART protocol. In fact, nearly all microcontrollers now come with a built-in USART interface, so the microcontroller side of things would be literally 'plug and play'[28].

On the Android side, the microcontroller would appear as a serial communication device. The 'usb-serial-for-android' library[29] would make it trivial to communicate with the microcontroller from an Android application via a serial port[30].

Because the connection is wired, there would be minimal interference so the speed could potentially go as high as the maximum USB 2.0 bit rate[31] of $480\,\mathrm{Mbit\,s^{-1}}$.

However, there would be a major disadvantage to this approach: that the oscilloscope would have to be physically connected to the tablet. This means the tablet would not be able to be freely moved around while using the oscilloscope. While a traditional oscilloscope operates this way, allowing wireless connection would make the oscilloscope much more vertasile: for example, the relatively cheap oscilloscope hardware could be left permanently connected in a hard-to-access electronics project, and accessed wirelessly when needed through the Android tablet.

[27] Silicon Labs, *Single-chip USB to UART Bridge*.

[28] Various configuration words would have to be moved to registers to initialise USART communication at the start of the program, but beyond that there would simply be registers for writing data, reading data and checking if data is available to be read

[29] mik3y, *usb-serial-for-android*.

[30] Serial ports are a throwback to the times when computers had physical RS232 serial connections, over which data could be sent and received using a very simple protocol. Nowadays, serial ports are virtual ports that send and receive data over much more complicated protocols such as USB and Bluetooth.

[31] Baud rate figures are not officially available, but it's clear from the bit rate that it would be extremely high

WiFi

It would be far beyond the scope of this project to communicate just using a 2.4 GHz antenna, so instead a discrete WiFi module would be needed. These have traditionally been relatively expensive[32], however Texas Instruments has recently released the CC3000, a module much cheaper than the competition.[33]

If the initial project was being created on a custom PCB, then the CC3000 could simply be included as a component on the PCB. However, the prototype is being created on a breadboard, so a breakout board (a board that contains the CC3000 and an antenna and exposes a digital interface to the CC3000 via pins that can fit in a breadboard) is needed.

The cheapest, most widely available breakout board is Adafruit's 'Adafruit CC3000 WiFi Breakout with Onboard Ceramic Antenna'. Adafruit are a US-based company, but the board can be easily obtained in the UK via eBay. However, it costs £25.

In terms of complexity, using the CC3000 would be more complicated than using USB, but not insurmountably so. Libraries exist for both AVR and PIC microcontrollers to interface with the CC3000, and on the Android side it would simply require interfacing with a TCP socket[34] (not significantly more complex than communicating via a USB serial connection).

One advantage WiFi has over USB is that it offers a wireless connection, negating the main disadvantage of USB. In addition, it would be able to offer very fast data transfer rates like USB (also like USB, official baud rate figures are not available, but the latest standard[35] offers a bit rate of up to $600\,\mathrm{Mbit\,s^{-1}}$).

However, the main disadvantage is the price. Even with the CC3000, the cheapest available option, at £25 the WiFi module would make up a considerable amount of the total cost of the oscilloscope.

Bluetooth

Like with WiFi, discrete Bluetooth modules are readily available. These operate in a similar way to the USB CP2102 — the microcontroller communicates with the module via USART, and the Android device via a serial port.

One example of such a module would be a JY-MCU HC-06. Via eBay, these are available relatively cheaply (about £5).

[32] So much so that for a time in the mid-2000s it become very common for devices such as amateur robots to communicate with computers over the 2.4 GHz spectrum, but using a different protocol to WiFi. This author has not been able to find out why such devices were significantly cheaper than WiFi devices, but one suspects it is due to both the complexity of the WiFi protocol and licensing costs required to implement it.

[33] Benchoff, *Finally, TI is producing simple, cheap WiFi modules.*

[34] Again, TCP sockets are significantly outside the scope of this project, but they can essentially be thought of as virtual counterparts to physical sockets. A wire can be used to connect physical sockets together to allow communication between them, and a protocol such as TCP (the base protocol for the internet) does the same thing for virtual sockets.

[35] 802.11n, which is supported by most wifi devices nowadays

This means one key advantage of Bluetooth is simplicity. The inbuilt USART functionality of the microcontroller can be used on the microcontroller side, and a Bluetooth serial library used on the Android side. The heavy lifting of the Bluetooth communication is done by the HC-06 and software built into the Android device.

Bluetooth obviously offers a wirelesss connection, and should work up to about 10 m. The maximum baud rate supported is $1.3824\,\mathrm{Mbit\,s^{-1}}$, which is slower than both USB and WiFi but fast enough for our purposes (remember the oscilloscope takes 8-bit samples at 500 kHz so the sampling rate is only $4\,\mathrm{Mbit\,s^{-1}}$).

Chosen Solution

The required wired connection between the oscilloscope and Android device ruled out USB, leaving WiFi and Bluetooth.

Data transfer speeds are not a big issue in this use case, and Bluetooth's $1.3824\,\mathrm{Mbit\,s^{-1}}$ is certainly fast enough. This means the key differences between Bluetooth and WiFi are complexity and cost. Bluetooth wins both of these, so Bluetooth was chosen as the communication method between microcontroller and Android device.

Chapter 2

System Development

2.1 Subsystems

The system will be split up into a number of subsystems, shown in fig. 2.1. Each of these subsystems is described in more detail in the sections below, and a brief description is given immediately below.

- Two analogue input channels come into the system, and are amplified and offset by amplification circuitry (with the gain controlled by a central microcontroller).
- These channels are both sampled using an Analogue to Digital Converter, which feeds into the microcontroller.
- A number of digital inputs are also fed straight into the microcontroller
- Through a Bluetooth transceiver, an Android device interacts with the microcontroller, receiving samples of both the analogue and digital inputs and controlling things such as the sampling rate and amplifier gain.

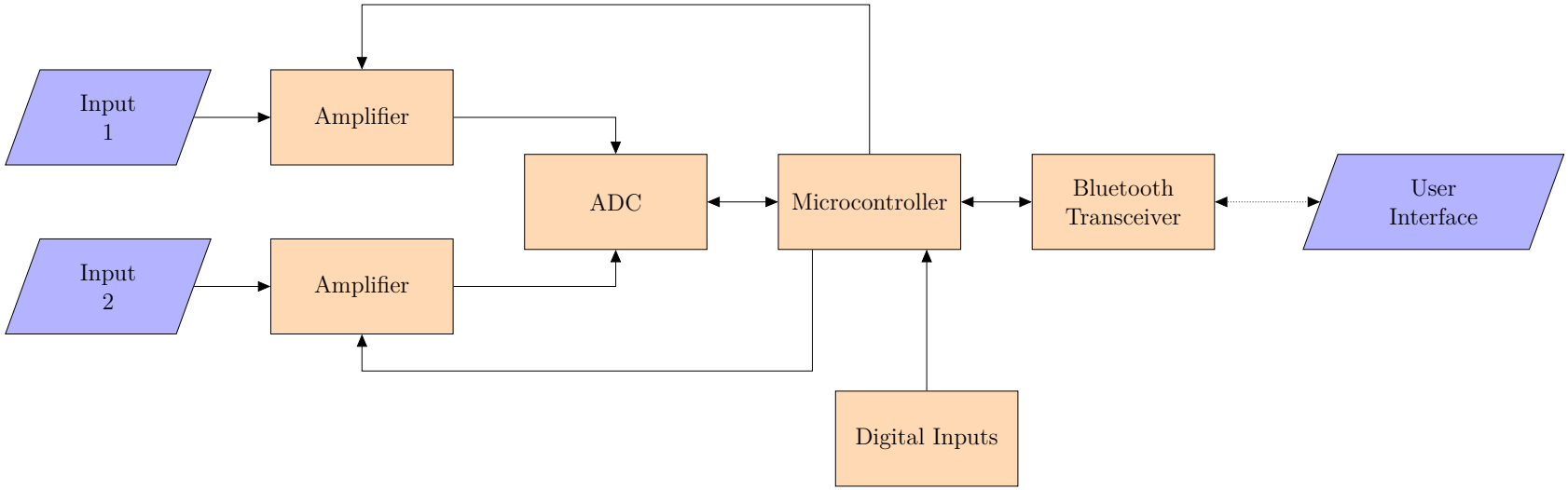

Figure 2.1: Subsystems within the oscilloscope

2.1.1 Analogue Inputs

For the purposes of this project wires are sufficient, however in a real-life scenario high-impedance, low-inductance test probes (usually coaxial cable) should be used.

2.1.2 Amplifiers

Each amplifier will have two purposes:

- To amplify the signal based on the Voltage Control of the oscilloscope. This is needed so the user can input a range of different voltage signals.
- To shift the signal so the minimum voltage is 0 V, rather than some negative voltage. This is required because the ADC will only accept an input greater than 0 V.

The chosen ADCs require an input voltage between 0 V and 5 V[1]. A non-inverting summing amplifier would not be suitable beause the offset added to the signal needs to remain constant while the gain changes, so instead the amplifier will be split into two stages.

The first stage will amplify the signal and offset it to be between 0 V and −5 V, below which the op-amp will saturate.

The second stage will invert this signal with a voltage gain of −1, meaning the output voltage will be between 0 V and 5 V as required.

First Stage

Circuit This stage will be implemented using a summing amplifier, with the inputs consisting of the input signal and a fixed voltage (for the sake of simplicity, the 5V supply wil be used).

See fig. 2.2 for a circuit diagram.

Component Calculations Suppose that the input signal varies from $-V_1$ to $+V_1$, and the amplifier has a gain of A and an offset of B. Then the output voltage V_o will be given by

$$-V_o = \frac{R_f}{R_1} V_i + \frac{R_f}{R_2} \cdot 5\,\text{V}$$

The first term will range from $-A \cdot V_1$ to $+A \cdot V_1$, and the second term will be B. So $-V_o$ will range from $B - A \cdot V_1$ to $B + A \cdot V_1$. Choosing a value of $B = 2.5\,\text{V}$ means that $-V_o$ will vary equally above and below 2.5 V, as we want[2](because the voltage must be between 0 V and 5 V, and 2.5 V is the midpoint of those two).

[1]The maximum voltage is actually V_{REF+}, which will be connected to V_{CC} for simplicity, which in this case is 5 V

[2]The op amp will saturate before outputting a voltage above 5 V, so we don't need to worry about this. If the gain or input voltage is sufficiently large, $-V_o$ could fall below 0 V. This means the next stage must saturate at 0 V and not output negative voltages, whatever the input.

R_f and R_2 will be fixed to give $\frac{R_f}{R_2} = 0.5$ to get the required offset. R_1 will be a digital potentiometer acting as a variable resistor, allowing the microcontroller to control the gain of the amplifier.

The input voltage range is from $\pm 50\,\text{mV}$ to $\pm 5\,\text{V}$, so the gain $A = \frac{R_f}{R_1}$ must vary from 0.5 to at least 50. Digital potentiometers are readily available in a number of values, so the standard $10\,\text{k}\Omega$ option will be chosen. This means the potentiometer can vary from the minimum resistance (usually $75\,\Omega$) to $10\,\text{k}\Omega$.

Using the values for A and R_1, this gives a value of $R_f = 5\,\text{k}\Omega$, meaning A will range from $\frac{5\,\text{k}\Omega}{10\,\text{k}\Omega} = 0.5$ to $\frac{5\,\text{k}\Omega}{75\,\Omega} \approx 66.7$, as required. In turn, this gives a value of $R_2 = 10\,\text{k}\Omega$.

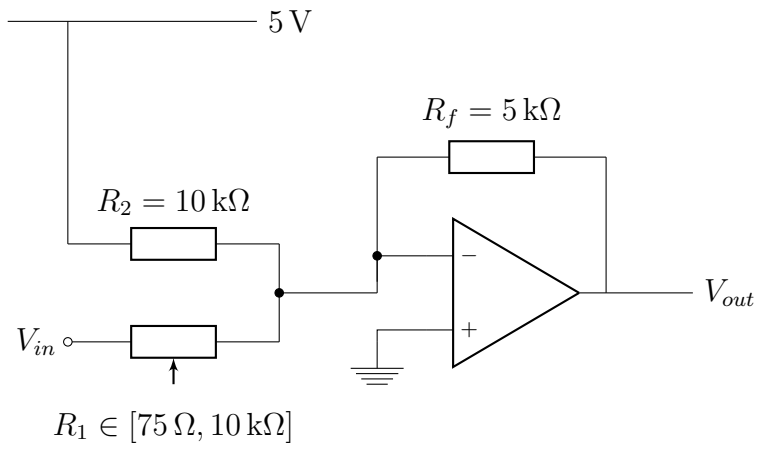

Figure 2.2: First stage amplifier circuit diagram

Component Selection The Microchip MCP4131-103E/P will be used for the digital potentiometer. It offers enough precision (typically 0.25%), offers a simple interface for changing the resistance (SPI) and is cheap and readily available in a PDIP package. Additionally, free samples are available from Microchip. To make the component calculations made earlier work, the $10\,\text{k}\Omega$ MCP4131-103 variety will be used.

The op-amps should be of rail-to-rail type (i.e. they can saturate at almost 0V and V_s) and be able to cope with $V_s = 5\,\text{V}$, $V_i = 5\,\text{V}$ and input frequencies of up to 1 MHz. The Analog Devices AD8031 meets all of these requirements (at 1 MHz it gives an open-loop gain of almost 25 dB) and is also cheap and readily available in PDIP packaging. Due to the availability of free samples, 2 AD8032ANZs (consisting of two AD8031 op amps in each chip) will be used.

Second Stage

This stage simply has to invert the previous stage, saturating at 0 V and 5 V. For this we can use a simple inverting amplifier.

$$V_{out} = \frac{R_f}{R_1} V_{in}$$

Choosing a standard value of $R_f = 10\,\text{k}\Omega$, we obtain $R_1 = 10\,\text{k}\Omega$.

As detailed in the previous section, AD8031 op amps will be used.

See fig. 2.3 for a circuit diagram.

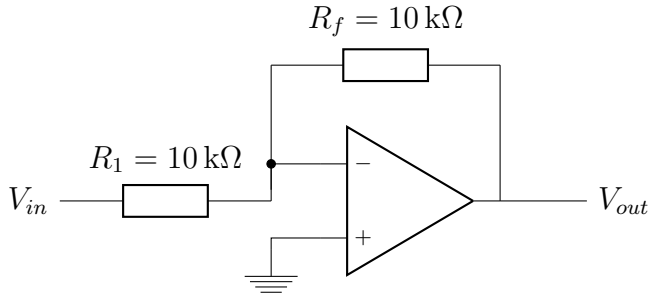

Figure 2.3: Second stage amplifier circuit diagram

Simulation

To check the frequency response of the amplification circuit, it was simulated using LT-Spice (see fig. 2.4).

To test the simulation, a low frequency sine wave was applied, and as expected the output was a sine wave of amplitude 2.5 V between 0 V and 2.5 V. The resultant waveform can be seen in fig. 2.5. A FFT was then applied to this waveform to obtain the frequency spectrums seen in fig. 2.6. As required, the spectrum of the output was identical to the spectrum of the input, with the slightly lower amplitudes being due to a lower wave amplitude (2.5 V c.f. 5 V).

A frequency sweep was then applied to produce a Bode plot. The details of such a plot are beyond the scope of this project, but let us just note that the gain is very consistent across all frequencies we'll be using (as predicted: the op amp has a stated bandwidth of 80 MHz, much higher than 1 MHz), and while it becomes almost 9° out of phase at very high frequencies, this is not an issue for this project[3].

[3]The phase is only important to the oscilloscope when comparing the phase of two signals. To see two signals on screen at the same time base, their frequencies must be close enough together that they're both out of phase by approximately the same amount. Hence their phase difference is unchanged

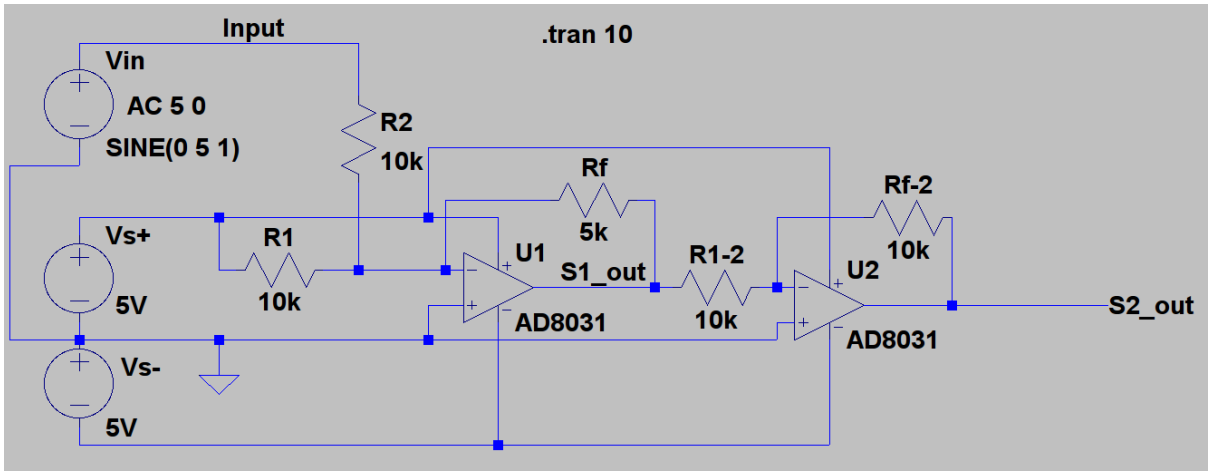

Figure 2.4: Amplification circuit in LTSpice

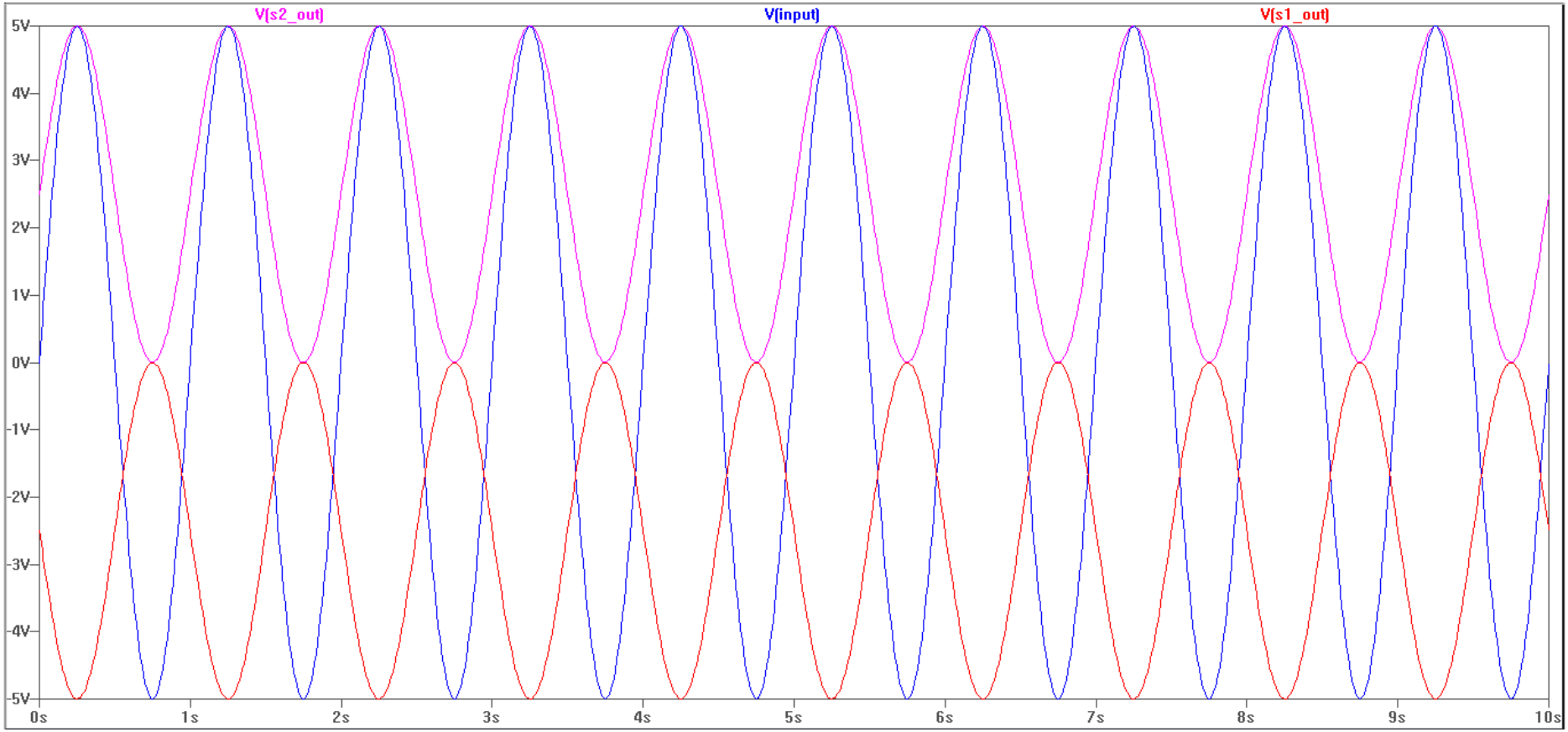

Figure 2.5: Waveform produced by amplification circuit at 1Hz (blue = original, red = first stage output, magenta = second stage output)

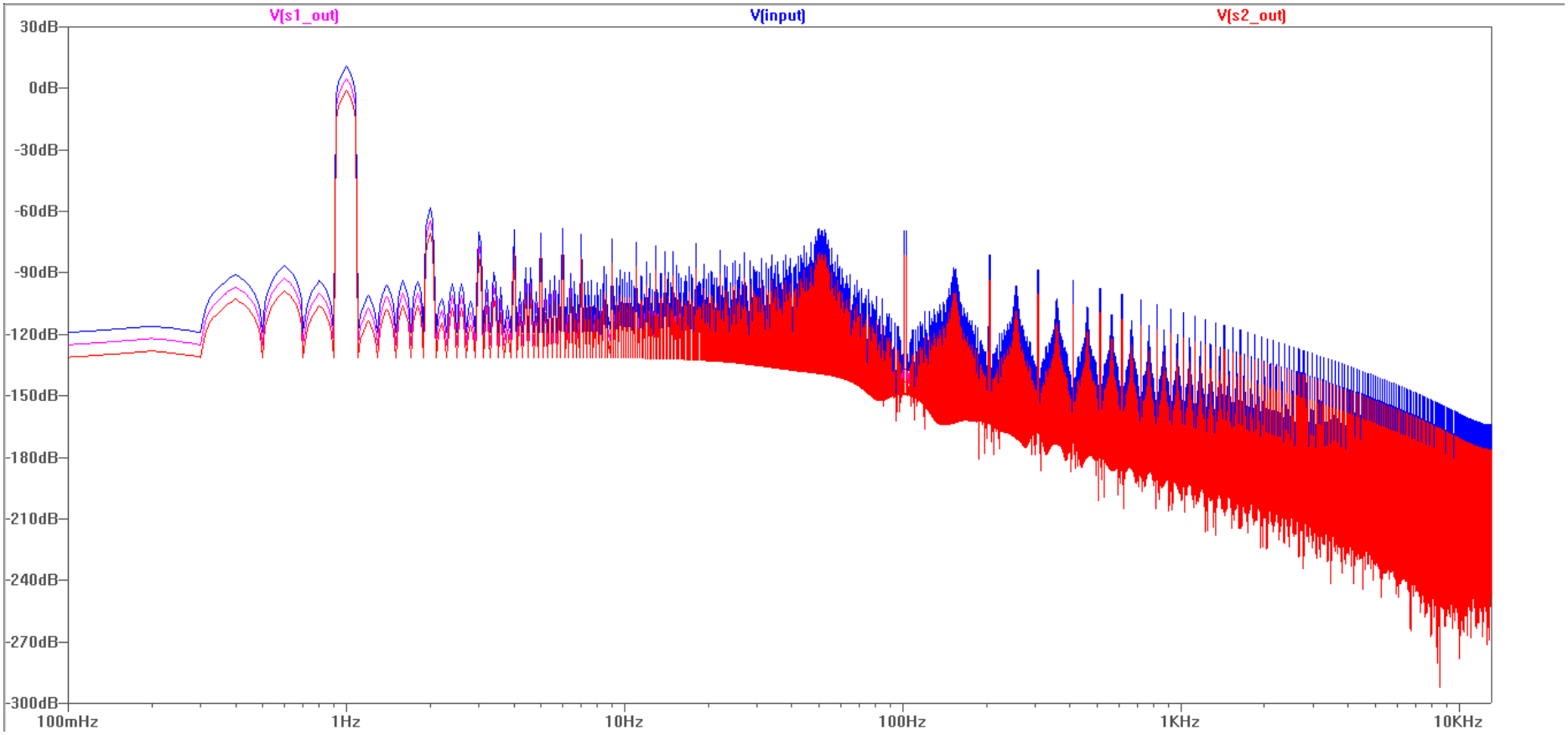

Figure 2.6: FFT of waveform produced by amplification circuit at 1Hz (red = original, blue = first stage output, magenta = second stage output)

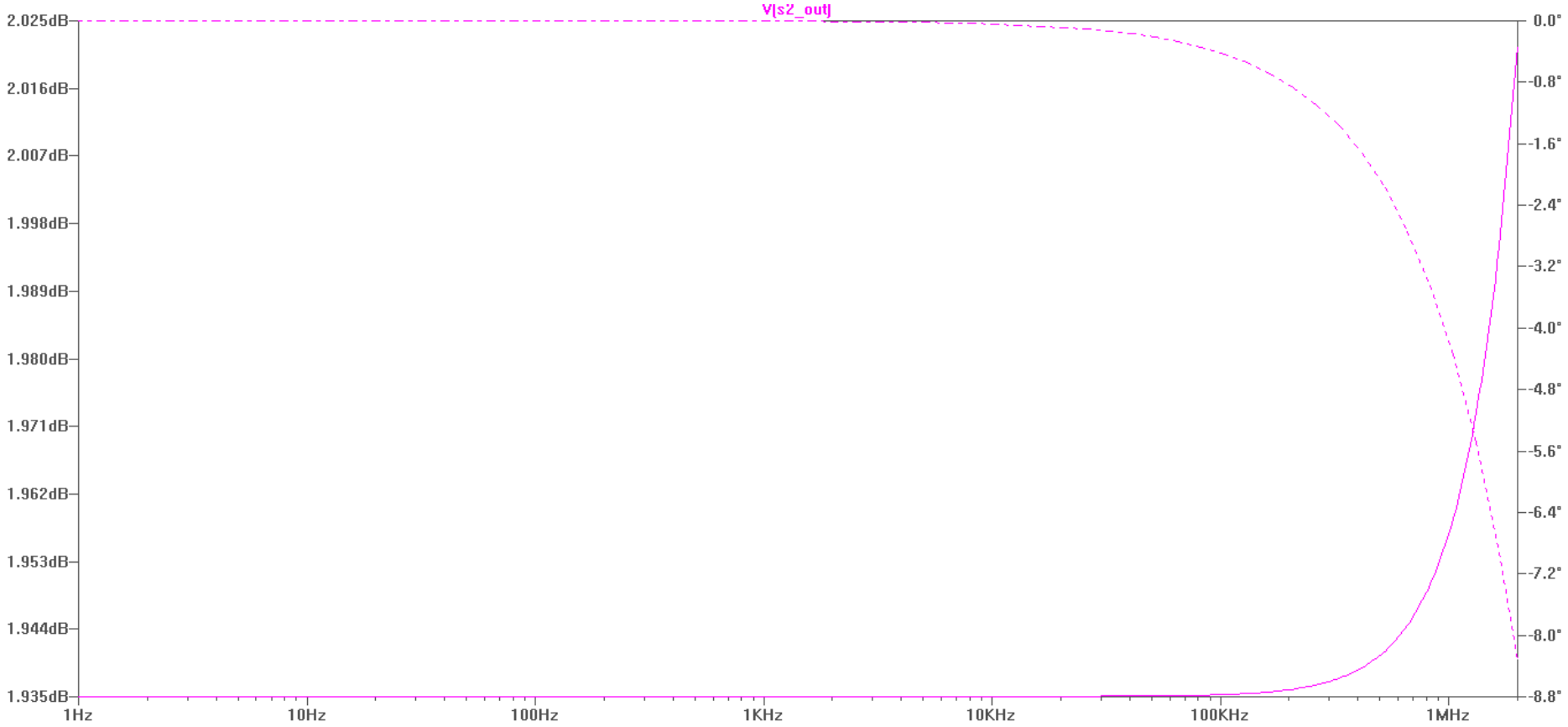

Figure 2.7: Frequency response of amplification circuit

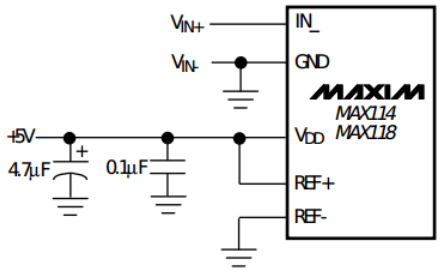

Figure 2.8: Power connections for the MAX114[6]

2.1.3 ADCs

As discussed before, these should be able to sample at up to 1 MHz. The only readily available ADCs that can do this in PDIP packaging and have free samples are the 8-bit half-flash[4] Maxim Integrated MAX114 ADCs.

There are a number of modes the ADCs can be operated in, however the pipelined mode offers a very good compromise between complexity and speed. The full details are in Maxim Integrated (*MAX114/MAX118 Datasheet*), but as the ADCs are operated from a microcontroller in which there's a delay of 62.5 ns (assuming a 16 MHz clock) between each instruction, the operating procedure can be simplified to the following:

1. Use A_0 and A_1 to choose the input signal (00 for input 0, 01 for input 1, the only two inputs used in this project)
2. Pull $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$[5] low
3. Wait approximately 250 ns
4. Pull $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$ back high
5. Read in a sample from D_0 through D_7

Taken directly from the datasheet, section 2.1.3 shows the connections that must be made to power the MAX114 and provide it with a reference voltage.

Additionally, $\overline{PWRDN}$ must be kept high to keep the ADC powered on, and $MODE$ must be kept high to keep the ADC in Read-Write mode (of which pipeline mode is a subset).

2.1.4 Microcontroller

Due to this author's significant previous experience with AVR microcontrollers, one will be used as there are no significant disadvantages over a PIC. The main requirements the microcontroller needs to satisfy are:

- Ability to use a 16 MHz clock
- Inbuilt UART communication (to communicate with the Bluetooth module)
- A significant number (at least 20) input/output pins

[4]Similar to flash ADCs, but requiring $\approx 2^{\frac{n}{2}}$ as opposed to $\approx 2^n$ parts

[5]For simplicity, $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$ can all be physically tied together and connected to just one microcontroller input

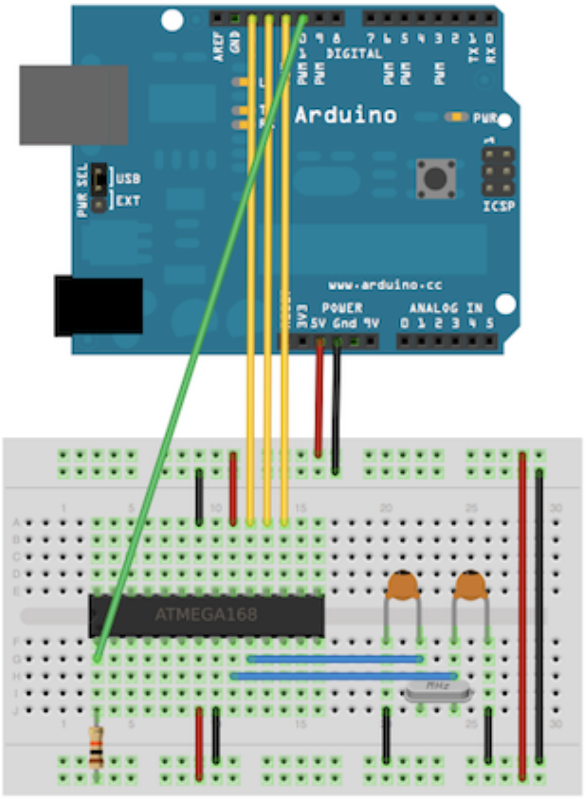

Figure 2.9: Breadboard diagram produced by Arduino (*ArduinoToBreadboard*) showing the programming and power connections between an Arduino and an ATMega328

- At least 1 kbyte of SRAM (to store the 8192 bits of samples)
- Availability in a PDIP package

The ATMega328 (more commonly known as the chip powering most Arduinos) was chosen because it satisfies all of these requirements and this author had a number of spare chips not being used.

Circuit

For the chip to operate, a 16 MHz crystal oscillator must be connected between $XTAL1$ and $XTAL$, with a 22 pF capacitor from each pin to ground. Additionally, the following power connections must be made:

- $RESET$ to 5 V through a 10 kΩ resistor
- Both GNDs to ground
- VCC and $AVCC$ to 5 V

To program the chip, an AVR programmer must be used. These are available very cheaply (such as the USBASP for £5 from eBay), but an Arduino (essentially a microcontroller development board) can also be used, as shown in section 2.1.4. As this author had a spare Arduino, this option was used.

Serial Interface

The following protocol was designed for serial communication between the Android device and the microcontroller. After taking a full 1024 samples, the microcontroller checks the serial buffer for one of the following commands and acts on it, responding via serial if necessary.

Traditionally, serial is a textual interface, however for efficiency's sake in this case it was used as a binary interface. Each transmitted character is stored as an 8-bit character, so the protocol is based on the integer values of those characters. For example, 65 below corresponds to a capital 'A'.

`0x00` Return all samples for the first analogue channel (1024 bytes, followed by a newline.[7])

`0x01` Return all samples for the second analogue channel (1024 bytes, followed by a newline)

`0x02` Wait for the next transmitted byte, X, and return all values for the Xth digital channel (1024 bits = 128 bytes, followed by a newline)

`0x03` Use analogue channel 1 to trigger (returns a newline)

`0x04` Use analogue channel 2 to trigger (returns a newline)

`0x05` Wait for the next transmitted byte, X, and use the Xth digital channel to trigger (returns a newline)

`0x06` Wait for the next two transmitted bytes, X and Y and use XY[8] as the number of microseconds to pause in between samples (returns a newline)

`0x07` Wait for the next transmitted byte, X, and use X as the analogue trigger threshold (returnsa newline)

`0x08` Wait for the next transmitted byte, X, and use the 7 least significant bits of X as the digital pot value in an amplifier (if the most significant bit of X is high, the second amplifier circuit, otherwise the first amplifier circuit) (returns a newline)

`0x09` Enable analogue channels 1 and 2 (this means slower sampling) (returns a newline)

`0x0A` Noop (this is the newline character) (returns a newline)

`0x0B` Enable analogue channel 1 only (faster sampling) sampling) (returns a newline)

So, for example, to receive samples for the second analogue channel the Android device would have to send `0x01`, then store the next 1024 received bytes.

ADC Communication

The way in which the microcontroller will communicate with the ADC is described in section 2.1.3.

[7] The character used to indicate a new line should start in text on a computer. Here, it's just equivalent to `0x0A`

[8] For example, if X = `0x50` and Y = `0x1A` then use `0x501A` as the delay

Digital Potentiometer Commmunication

The resistance of the digital potentiometers is set using an SPI interface. While the details can get more complicated, SPI (Serial Peripheral Interface) is essentially a very simple interface, consisting of four pins: $SCLK$, $MOSI$, $MISO$ and $\overline{SS}$.

There are two devices: a master and a slave. $SCLK$ is simply a clock output from master. On either the rising or falling edge of $SCLK$, a bit of data is transmitted from master to slave on $MOSI$ and slave to master on $MISO$. $\overline{SS}$ is simply an active low slave select (i.e. the slave only communicates when $\overline{SS}$ is low).

There are four different SPI modes, of which the digital potentiometers support two ($CPOL = 0 = CPHA$ and $CPOL = 1 = CPHA$). In both modes, data is captured on the rising edge and propogated on the falling edge. The difference is simply the first value of the clock: in the first mode, it's low, and in the second it's high.

The maximum clock supported by the digital potentiomers is 10 MHz, which is faster than we can reliably communicate at with a 16 MHz microcontroller anyway.

To send a 16-bit byte to the digital potentiometer the following must be done:

1. Pull $\overline{SS}$ low
2. Wait one $SCLK$ period
3. Starting from a high (an artbitrary choice), pulse $SCLK$
4. On each falling edge, send a bit of data until 16 bits have been sent
5. Pull $SCLK$ back high
6. Wait one clock period
7. Pull $\overline{SS}$ back high

More on digital pot. communication

Program

To perform all of the above tasks, then, the microcontroller will need to run the following program:

- Loop forever:
  - Loop 1024 times:
    - Take sample from first analog channel
    - Take sample from second analog channel, if enabled
    - Take 1-bit sample from each digital channel
  - Look at most recent serial command, and perform any actions and send any data it specifies
  - Clear most recent serial command
- Interrupt on serial byte received:
  - Store as most recent serial command
  - Receive up to 2 more bytes depending on the command

2.1.5 Bluetooth Transceiver

As discussed in section 1.5, Bluetooth was chosen to communicate with the Android device acting as the user interface. In particular, the JY-MCU HC-06 was chosen.

2.1.6 User Interface

A high-quality enough screen that interfaced directly with the microcontroller would be quite expensive, so instead an Android device was chosen as the user interface.

Android was chosen over the alternatives (iOS, etc) primarily because it's the most popular mobile operating system, allowing the greatest number of people to use the oscilliscope. Additionally, it allows easy and open development, whereas competitors such as iOS require costly developer licenses, proprietary software and specific hardware.

List of Corrections

Bibliography

[1] Arduino. *ArduinoToBreadboard*. accessed 26/01/14. URL: http://arduino.cc/en/Tutorial/ArduinoToBreadboard.

[2] Brian Benchoff. *Finally, TI is producing simple, cheap WiFi modules*. accessed 26/01/14. 2013. URL: http://hackaday.com/2013/01/12/finally-ti-is-producing-simple-cheap-wifi-modules/.

[3] Bitscope. *Bitscope Logic Analyzer*. accessed 26/01/14. URL: http://www.bitscope.com/software/logic/06.png.

[4] Gabotronics. *Digital oscilloscopes for hobbyists*. accessed 02/01/14. 2013. URL: http://www.gabotronics.com/resources/hobbyists-oscilloscopes.htm.

[5] Maxim Integrated. *MAX114/MAX118 Datasheet*. accessed 02/01/14. URL: http://datasheets.maximintegrated.com/en/ds/MAX114-MAX118.pdf.

[6] mik3y. *usb-serial-for-android*. accessed 26/01/14. URL: https://github.com/mik3y/usb-serial-for-android.

[7] W. Phillips. *Using an Oscilloscope*. accessed 26/01/14. URL: http://www.doctronics.co.uk/scope.htm.

[8] Picotech. *Advanced Triggering with PicoScope*. accessed 02/01/14. 2014. URL: http://www.picotech.com/education/oscilloscopes/advanced-triggering.html.

[9] Julian Roy. *Real-time processing system for dual-comb spectroscopy*. accessed 26/01/14. 2013. URL: http://nutaq.com/en/blog/real-time-processing-system-dual-comb-spectroscopy-part-1.

[10] Claude Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27 (1948), pp. 379–423, 623–656. URL: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

[11] Silicon Labs. *Single-chip USB to UART Bridge*. accessed 26/01/14. URL: http://www.silabs.com/Support%20Documents/TechnicalDocs/CP2102-9.pdf.

[12] Deepak Kumar Tala. *ASIC World*. accessed 01/02/14. URL: http://www.asic-world.com/verilog/index.html.

[13] TiePie Engineering. *Digital Data Acquisition*. accessed 26/01/14. URL: http://www.tiepie.com/en/classroom/Measurement_basics/Digital_Data_Acquisition.

[14] Alan Tong. *What to Look for When Choosing an Oscilloscope*. accessed 02/01/14. 2014. URL: http://www.picotech.com/applications/oscilloscope_tutorial.html.

[15] Wikipedia. *Verilog — Wikipedia, The Free Encyclopedia*. accessed 01/02/14. 2013. URL: http://en.wikipedia.org/wiki/Verilog.

[16] Joel Woodward. *What's the Difference Betwen a Mixed-Signal Oscilloscope And A Logic Analyzer?* accessed 26/01/14. 2012. URL: http://electronicdesign.com/test-amp-measurement/what-s-difference-between-mixed-signal-oscilloscope-and-logic-analyzer.