

ELECTRONICS A2 PROJECT

Bioniscope

Author:
Harry RICKARDS



2014

Contents

1	Problem Analysis and Solution Design	3
1.1	Problem Definition	3
1.2	Research into Existing Oscilloscopes	5
1.3	Practical Investigations	6
1.3.1	Oscilloscope Reliability Investigation	6
1.3.2	Bluetooth Distance Test	12
1.4	Numerical Parameters	15
1.5	Communication with Android Device	16
2	System Development	19
2.1	Subsystems	19
2.1.1	Analogue Inputs	21
2.1.2	Amplifiers	21
2.1.3	ADCs	27
2.1.4	Microcontroller	27
2.1.5	Bluetooth Transceiver	31
2.1.6	User Interface	31
2.2	Construction	31
2.2.1	PCB Layout	31
2.2.2	PCB Production	32
2.3	Enclosure	33
A	Fourier Series of a Square Wave	34

List of Figures

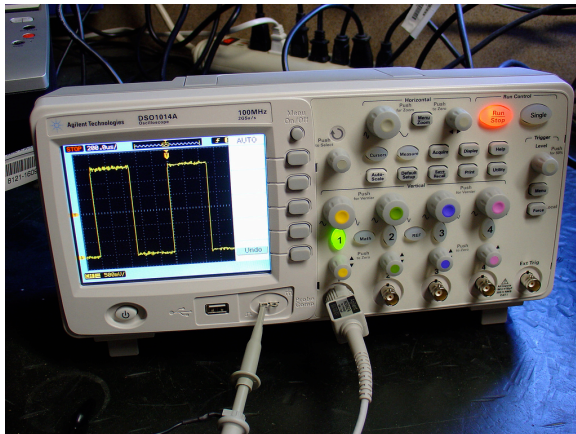
1.1	Existing oscilloscopes ¹	3
1.2	Logic Analyser Example Output	4
1.3	Equivalent Time Sampling Diagram	6
1.4	Equipment setup to test the CRO	8
1.5	Not-so-square wave shown on the oscilloscope	10
1.6	Approximations to a square wave up to certain harmonics	11
1.7	The Arduino code used to echo back any characters received via Bluetooth	13
1.8	The python code used to measure the accuracy of the Bluetooth link at various distances	14
2.1	Oscilloscope Subsystems	20
2.2	1 st Amplifier Stage	22
2.3	2 nd Amplifier Stage	23
2.4	Amplification Circuit Simulation	23
2.5	Amplification Circuit Waveform	24
2.6	Amplification Circuit Frequency Spectrum	25
2.7	Amplification Circuit Bode Plot	26
2.8	ADC Power Connections	27
2.9	Arduino as a Programmer	28
A.1	Square wave with amplitude 1 and frequency 1 Hz	34

¹jepoirrier, *Oscilloscope with noise*; mightyohm, *New Oscilloscope*.

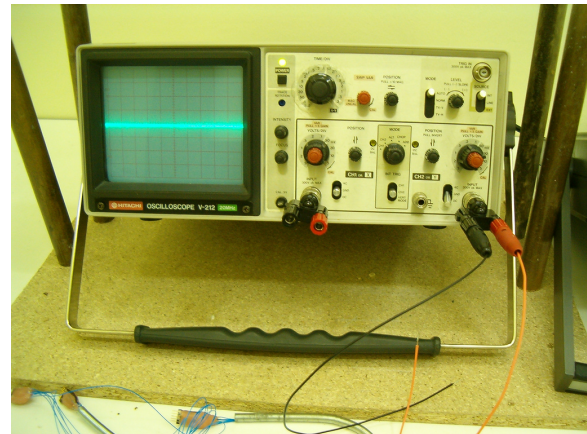
Problem Analysis and Solution Design

1.1 Problem Definition

CROs and DSOs



(a) Cathode Ray Oscilloscope



(b) Digital Storage Oscilloscope

Figure 1.1: Existing oscilloscopes¹

Oscilloscopes available in secondary schools or sixth forms are generally basic analogue oscilloscopes known as CROs, or Cathode Ray Oscilloscopes (named so because the display is a Cathode Ray Tube).²

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 FPGA³ or a DSP⁴), before being output onto some sort of digital screen.⁵

²Phillips, *Using an Oscilloscope*.

³**Field Programmable Gate Array** a device that can be programmed to ~become a very large logic circuit. This makes it much faster than a microcontroller at doing a specific job.

⁴**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 separate two audio signals)

⁵TiePie Engineering, *Digital Data Acquisition*.

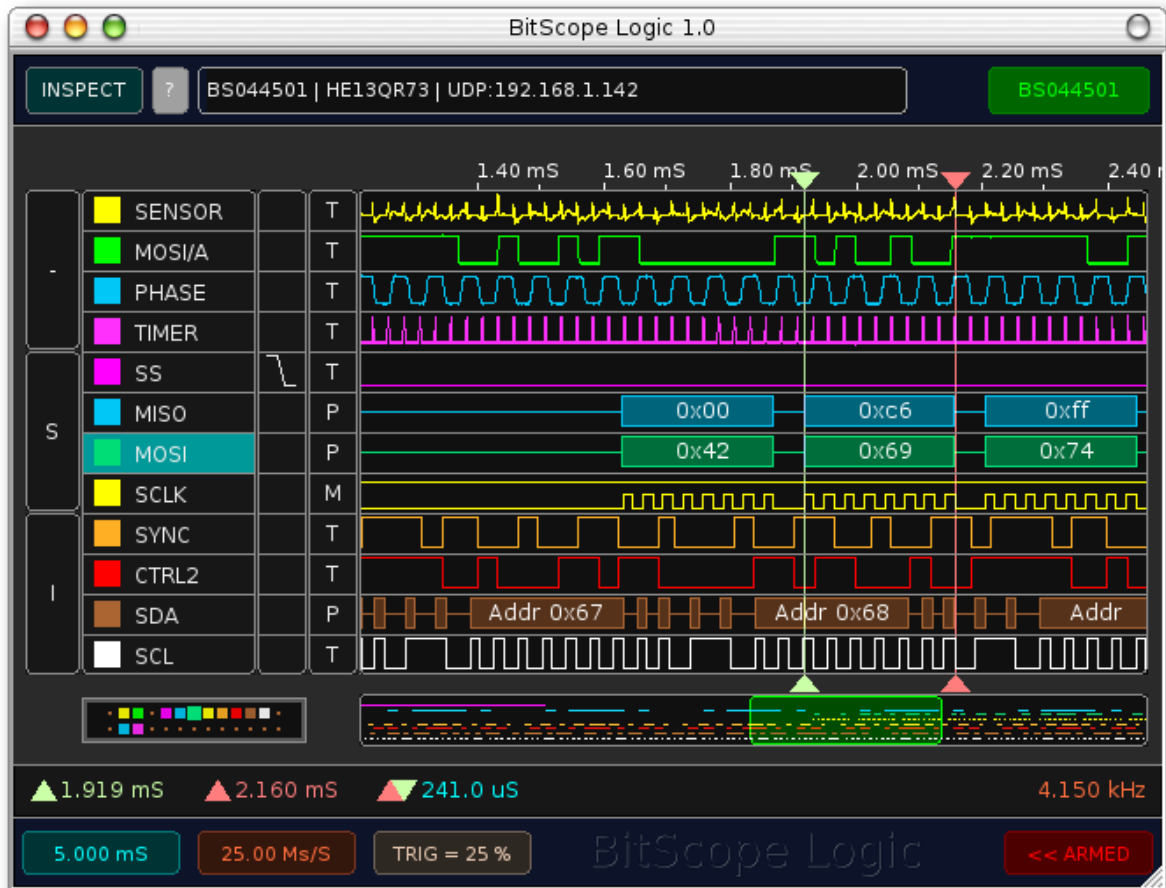


Figure 1.2: Example output from a Logic Analyser¹⁰

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)⁶ can be run to produce a frequency spectrum of the input signal, allowing the user to see both time-domain and frequency-domain representations of the signal.⁷

Additionally, a DSO can be used to create a basic Logic Analyser. This is essentially the digital equivalent of an oscilloscope — a relatively large number of digital wires and buses are connected to the Logic Analyser and a timing diagram is output^{8,9}.

⁶While DFTs are the 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 more quickly whilst achieving a similar result

⁷This 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

⁸In more advanced, standalone Logic Analysers, more complicated outputs such as decoded Ethernet packets can be output

⁹Woodward, *What's the Difference Between a Mixed-Signal Oscilloscope And A Logic Analyzer?*

Project Definition

This project¹¹ 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.

1.2 Research into Existing Oscilloscopes

Using two articles from Tong¹² and Gabotronics,¹³ digital oscilloscopes can be categorised by the following criteria:

- **Bandwidth** The bandwidth¹⁴ 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 fig. 1.3.

While equivalent-time sampling offers a much faster sampling rate, it is much more complicated to implement.

- **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 triggering¹⁵ (when the signal passes a threshold on a rising edge¹⁶). There are also more advanced triggers such

¹¹As seen in various places in this document, the project was nicknamed *Bioniscope* — a portmanteau of bionic (from Android) and oscilloscope

¹²Tong, *What to Look for When Choosing an Oscilloscope*.

¹³Gabotronics, *Digital oscilloscopes for hobbyists*.

¹⁴Most commonly, bandwidth is taken to be the range of frequencies where the signal has an amplitude gain of $-3\text{ dB} \approx 71\%$

¹⁵Picotech, *Advanced Triggering with PicoScope*.

¹⁶A falling edge could also be used, but the type of edge is part of the trigger specifications and stays constant

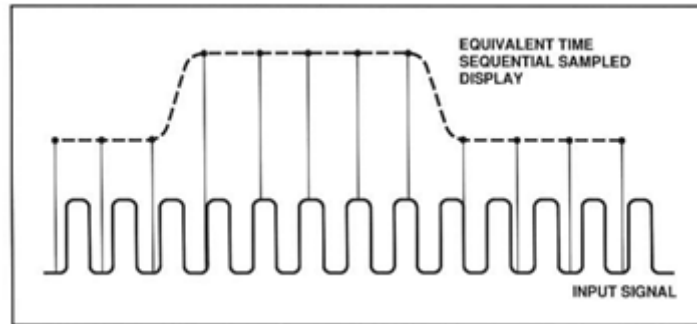


Figure 1.3: Diagram illustrating equivalent time sampling¹⁹

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/IP¹⁷ and HTTP¹⁸ packets from a digital 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.

1.3 Practical Investigations

1.3.1 Oscilloscope Reliability Investigation

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

An FPGA²⁰ 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 project²¹, let us just say that Verilog is a way of describing the propagation of signals based on dependencies on both time and other signals.²² See listing 1 for a very simple Verilog example.

```
always @(posedge) clk begin
    q <= d;
end
```

Listing 1: A simple flip-flop implemented in Verilog

¹⁷The protocol used for internet networking

¹⁸The protocol used to transfer web pages over the internet

²⁰**Field Programmable Gate Array** a device that can be programmed to ~become a very large logic circuit. This makes it much faster than a microcontroller at doing a specific job.

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

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

The FPGA was programmed to perform 50% duty-cycle clock divisions of the FPGA's internal²³ 50 MHz clock, meaning the output frequency could be easily calculated using

$$f = \frac{50 \text{ MHz}}{2^n} \quad (1.1)$$

where n is the clock divisor. The accuracy of the crystal oscillator is so high (an average error value would be around 0.0005%²⁴) compared to the accuracy of reading values from a small oscilloscope screen that we can ignore it in this instance.

The Verilog code used can be seen in listing 2.

```
// Output a square wave signal of various frequencies to test a CRO
module oscilloscope_test(clk, signal);
    // Accurate 50MHz clock input signal
    input clk;

    // Output square wave
    output wire signal;

    // Module that divides a frequency to produce a 50% duty-cycle output
    // using flip-flops
    clock_divider d1 (
        .clk_in(clk),
        .clk_out(signal)
    );
    // clk is divided by 2^divisor
    // So increase by 1 to halve the output frequency
    defparam d1.divisor = 21;
endmodule
// Divide a clock by 2^divisor with a 50% duty-cycle output (1:1 mark:space)
module clock_divider(clk_in, clk_out);
    // This is intended to be changed by the code calling this module
    parameter divisor = 2;

    // Input to be divided
    input clk_in;

    // Create a counter out of flip flops, and toggle clk_bit whenever the
    // counter's most significant bit toggles. This produces a 50% duty-cycle
    // output that we use to toggle clk_out.
    reg [divisor-1:0] counter = 0;
    always @(posedge clk_in) counter <= counter + 1;
    output reg clk_out = counter[divisor-1];
endmodule
```

Listing 2: Code used to produce the square wave to test the oscilloscope

²³A high-quality crystal oscillator

²⁴Semtech, *Improving the Accuracy of a Crystal Oscillator*.

Data

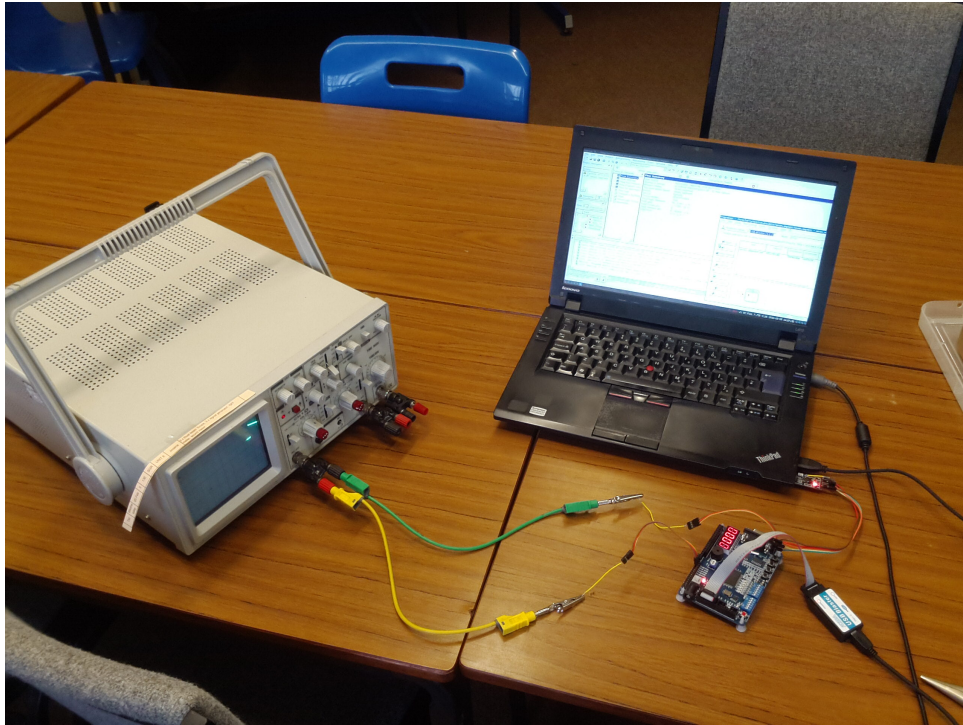


Figure 1.4: Equipment setup to test the CRO

Seen in fig. 1.4, the equipment was setup as described above. Frequency readings were taken over a number of values of n ranging from 4 to 19. For each reading, the horizontal and vertical division settings were recorded, as well as the number of divisions counted in one period of the wave. From these, the measured frequency and amplitude were calculated.

These measured values were compared to the actual frequency (as calculated by ??) and amplitude (the digital output from the FPGA ranged from 0 V to its logic-high level of 3.3 V, leading to an amplitude of 1.65 V) to find error values for the frequency and amplitude.

The numerical results of the experiment can be seen in table 1.1.

With $n = 19$, the frequency was low enough that it was very difficult to take a reading of the number of y divisions per period due to the nature of cathode ray tubes. This can be seen in the results: the frequency error jumps from less than 1% to over 90%. Because of this, $n = 19$ will be considered an anomalous result and disregarded for the rest of this analysis.

Note the high level of accuracy, particularly with regards to frequency. Due to previous bad experiences with this oscilloscope, your author was rather surprised at the results. Almost all of the error can actually be put down to errors in the human measurement process. Take $n = 7$ as an example: the number of X divisions was measured as 5.1 to 1 decimal place, giving a percentage error of 0.98%. It is purely coincidental then, that the actual error is less than half that at 0.39%!

n	Time/div (μs)	Voltage/div (V)	Measured				Actual		Error (%)	
			X div	Y div	Frequency (kHz)	Amplitude (V)	Frequency (kHz)	Amplitude (V)	Frequency	Amplitude
4	0.1	2	3.4	2.0	2940	2.0	3120	1.65	5.88	21.21
5	0.2	2	3.1	1.8	1610	1.8	1560	1.65	3.23	9.09
6	0.5	2	2.5	1.8	800	1.8	781	1.65	2.40	9.09
7	0.5	2	5.1	1.7	392	1.7	391	1.65	0.39	3.03
8	1	2	5.1	1.7	196	1.7	195	1.65	0.39	3.03
9	2	2	5.1	1.7	98.0	1.7	97.7	1.65	0.39	3.03
10	5	2	4.1	1.7	48.8	1.7	48.8	1.65	0.10	3.03
12	20	2	4.1	1.7	12.2	1.7	12.2	1.65	0.10	3.03
14	100	2	3.2	1.7	3.12	1.7	3.05	1.65	2.40	3.03
16	500	2	2.6	1.8	0.769	1.8	0.763	1.65	0.82	9.09
19	2000	2	2.7	1.7	0.185	1.7	0.0954	1.65	94.18	3.03

Table 1.1: Results from oscilloscope accuracy experiment

Qualitative Results

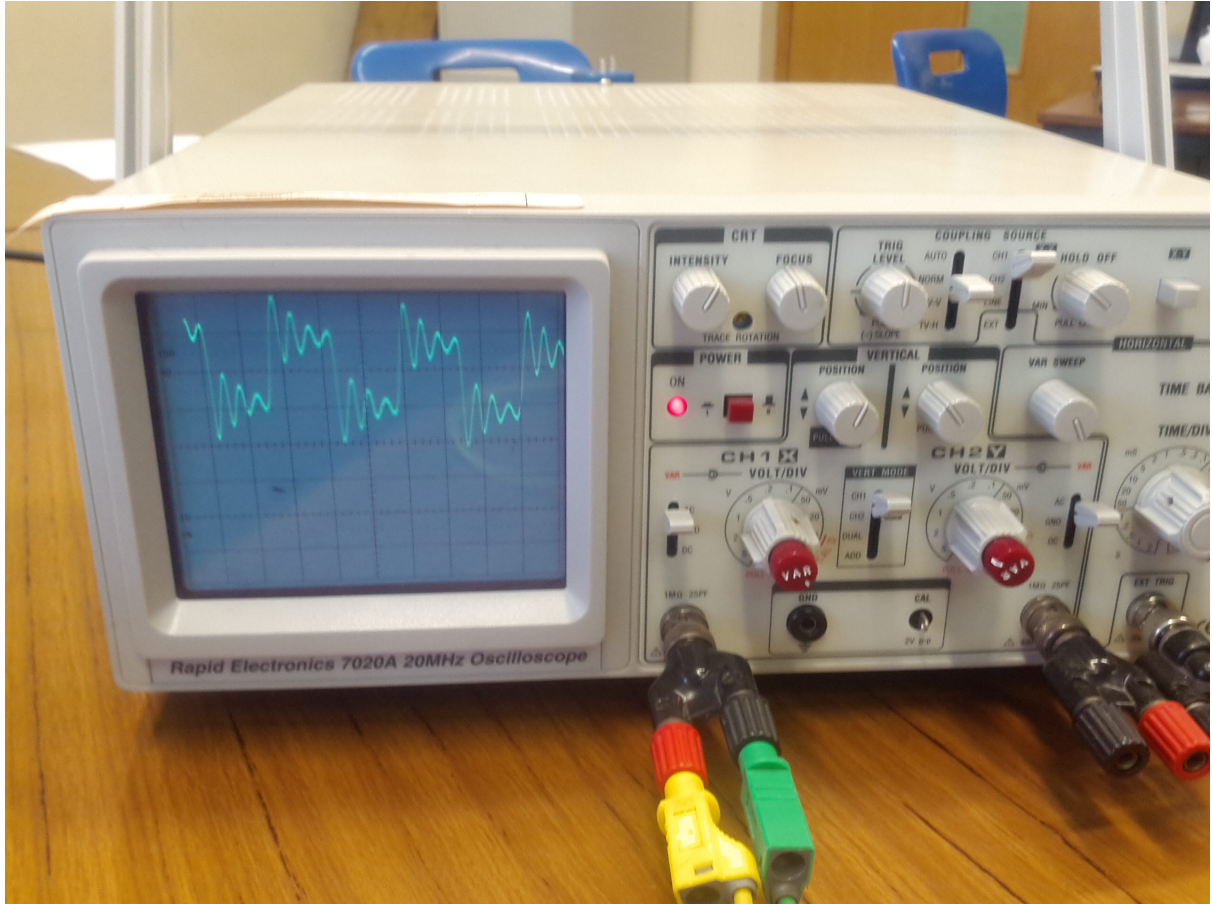


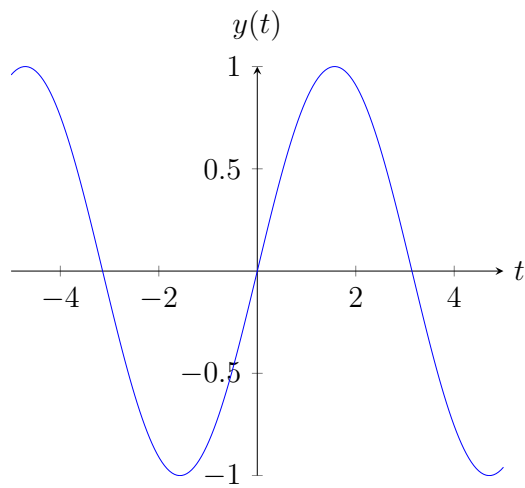
Figure 1.5: Not-so-square wave shown on the oscilloscope

In qualitative terms, however, the oscilloscope was not nearly as accurate. Looking at fig. 1.5, one can see that at high frequencies the analogue bandwidth of the oscilloscope comes into play, causing the digital input to look very much like a summation of sine waves. In fact, this was still visible to a lesser extent until $n = 12$ (corresponding to a frequency of 12.2 kHz). This means that while the oscilloscope can be used to accurately determine the frequency of high-frequency signals, it can't be used to determine the shape of a signal with any accuracy.

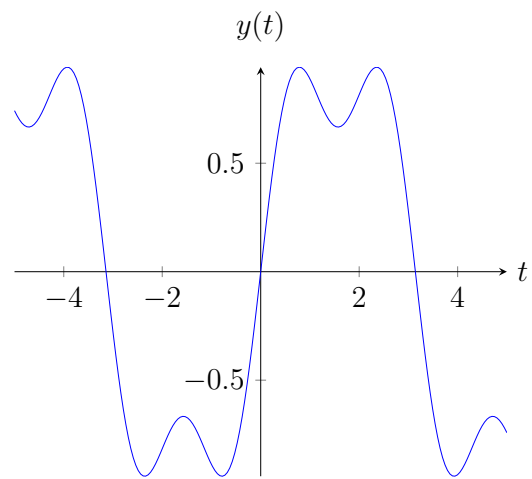
Using the maths detailed in appendix A we can plot graphs of square wave approximations up to a certain number of harmonics, as seen in fig. 1.6. Comparing these to fig. 1.5, we find that the closest match²⁵ is the seventh harmonic graph. This means we can approximate the analogue bandwidth of the oscilloscope as 7 times the frequency of the signal, or $7 \times 3.12 \text{ MHz} = 21.8 \text{ MHz}$ ²⁶.

²⁵The match is not exact because harmonics above the seventh harmonic are present, they just have a reduced amplitude. However by looking at the number of peaks in one period of a wave we find the seventh harmonic is the highest one with a major component present.

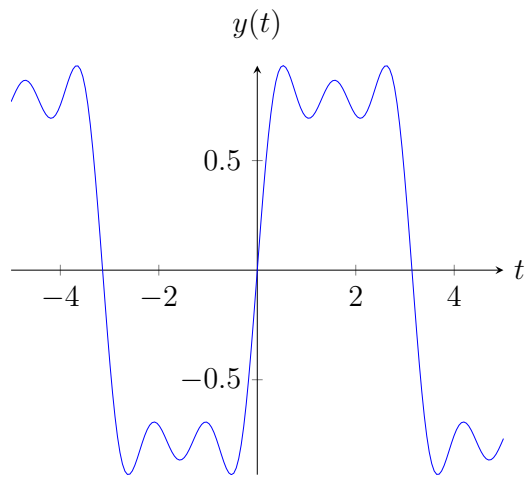
²⁶Checking the datasheet of the oscilloscope, we find the quoted bandwidth is 20 MHz meaning we are accurate to within a third of a harmonic, as close as we could hope using this basic method.



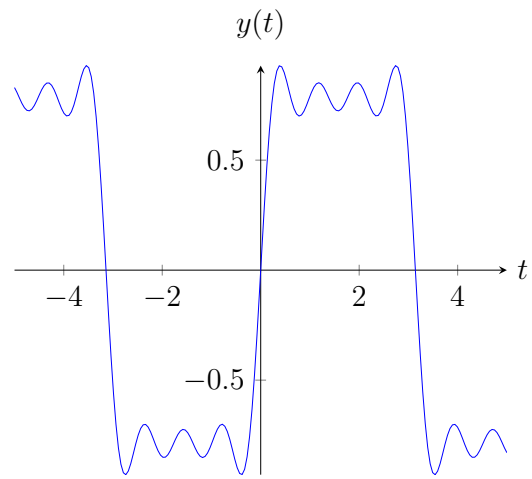
(a) First Harmonic



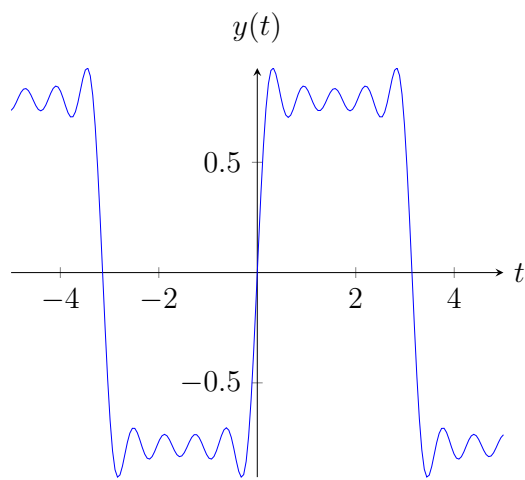
(b) Third Harmonic



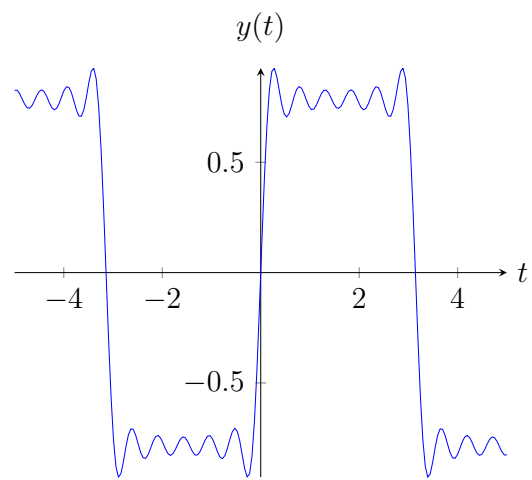
(c) Fifth Harmonic



(d) Seventh Harmonic



(e) Ninth Harmonic



(f) Eleventh Harmonic

Figure 1.6: Approximations to a square wave up to certain harmonics

Distance (m)	Reliability (%)
0.0	100.00
1.0	100.00
2.0	100.00
3.0	100.00
4.0	100.00
5.0	100.00
5.5	6.51
6.0	0.00

Table 1.2: Results from bluetooth distance experiment

1.3.2 Bluetooth Distance Test

As discussed in section 1.5, Bluetooth will be used to communicate between the microcontroller and Android device. Because of this, a practical investigation was undertaken to test the reliability of the Bluetooth link at different distances.

The Bluetooth module to eventually be used in the final product, an HC06, was connected to an Arduino (an ATmega microcontroller development board allowing quick prototyping) and programmed to echo any data sent to it (so if the character *A* was received by the HC06, it would immediately transmit back the character *A*). This code can be seen in fig. 1.7.

A python²⁷ program was then written on a laptop computer (with an inbuilt Bluetooth module) to communicate with the HC06. 384²⁸ bytes were sent out, and the received data compared to the sent data to produce an accuracy figure. This code can be seen in fig. 1.8²⁹.

This program was run at with the laptop a number of different distances from the HC06. The results are shown in table 1.2.

At distances up to and including 5 m, the link was 100% reliable. At 5.5 m that figure dropped dramatically (suggesting that the connection was lost partway through the test), and at further distances the link dropped out completely. This means that the oscilloscope should be able to be used at distances of up to 5 m away from the device.



²⁷A very high-level computer programming language that's suited for tasks such as this, with libraries for things such as serial communication. For more information see various online resources.

²⁸3 times 128, and a convenient compromise between speed and accuracy

²⁹For details on how the Bluetooth receiver in the laptop was setup as a serial device, see the online page for this project at <http://github.com/hrickards/bioniscope>.

```

// IMPORTANT
// Delays need to be added into setup() and loop() if the Arduino is to be
// programmed over USB. If the code is left as it is, the Arduino will not
// be able to communicate with the IDE over USB, so a dedicated AVR programmer
// (or another Arduino) will be needed to upload new sketches.

// Run when the Arduino starts up initially
void setup() {
    // Begin serial communication with the HC06 at a baud rate of 9600
    Serial.begin(9600);

    // Use pin 13 (connected to an LED) to show visually when data is being
    // written
    pinMode(13, OUTPUT);
}

// Loops continuously
void loop() {
    // Turn LED off
    digitalWrite(13, LOW);

    // Wait til serial available
    while (!Serial.available());

    // Turn LED on
    digitalWrite(13, HIGH);

    // Echo back the received serial byte
    Serial.write(Serial.read());
}

```

Figure 1.7: The Arduino code used to echo back any characters received via Bluetooth


```

import serial # pySerial used for serial communication with BT receiver

# Number of bytes of data to send each time
ITERATIONS = 384

# Open a serial socket to the BT device
# If this line fails, check /dev/rfcomm0 is the right port
ser = serial.Serial('/dev/rfcomm0', 9600, timeout=1)

# Remove any initial noise by writing a newline and waiting for it to return
ser.write("\n")
ser.readline()

# Write results to data.txt
f = open("data.txt", "wb")

j = 0 # Number of times the test has been run

# Keep running the indented part until the program is stopped
while True:
    # To be filled with booleans that are true iff the data is echoed correctly
    data = []
    # The character to send and have echoed back. Will be changed by the code
    # below.
    char = 0x00

    # Do the indented part ITERATIONS number of times
    for i in range(ITERATIONS):
        # Increment char, wrapping back to 0x00 after 0xFF
        char = (char + 1) % 0xFF
        # Write the character
        ser.write(chr(char))
        # Append to data whether or not the character was echoed correctly
        data.append(ser.read() == chr(char))

    # The line of output to show to the user and write to data.txt
    # test number: reliability average
    output = "%d: %f" % (j, sum(data)*100.0/len(data))
    print(output)
    f.write(output + "\n")

    j += 1
    raw_input("...") # Wait for the user to press enter

ser.close()
f.close()

```

Figure 1.8: The python code used to measure the accuracy of the Bluetooth link at various distances

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 frequency³⁰ 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 DIP³¹ package (suitable for breadboards) are not readily available beyond 2 MHz. Furthermore, the only 2 MHz DIP ADC, the AD7822, is renowned for being particularly 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). The effect of this can be seen all too clearly in fig. 1.5 To be safe, we will specify a minimum bandwidth of 1 MHz.

Resolution

Inaccuracies in the oscilloscope 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.

³⁰Actually, 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

³¹Electronics 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.

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 V³².

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 CP2102,³³ 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'³⁴.

On the Android side, the microcontroller would appear as a serial communication device. The 'usb-serial-for-android' library³⁵ would make it trivial to communicate with

³²While 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 the supply voltage.

³³Silicon Labs, *Single-chip USB to UART Bridge*.

³⁴Various 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

³⁵mik3y, *usb-serial-for-android*.

the microcontroller from an Android application via a serial port³⁶.

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 rate³⁷ of 480 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 versatile: 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.

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 expensive³⁸, however Texas Instruments has recently released the CC3000, a module much cheaper than the competition.³⁹

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 socket⁴⁰ (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

³⁶Serial 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.

³⁷Baud rate figures are not officially available, but it's clear from the bit rate that it would be extremely high

³⁸So much so that for a time in the mid-2000s it became 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.

³⁹Benchoff, *Finally, TI is producing simple, cheap WiFi modules.*

⁴⁰Again, 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.

rates like USB (also like USB, official baud rate figures are not available, but the latest standard⁴¹ offers a bit rate of up to 600 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).

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 wireless connection, and should work up to about 10 m. The maximum baud rate supported is 1.3824 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 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 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.

⁴¹802.11n, which is supported by most wifi devices nowadays

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.

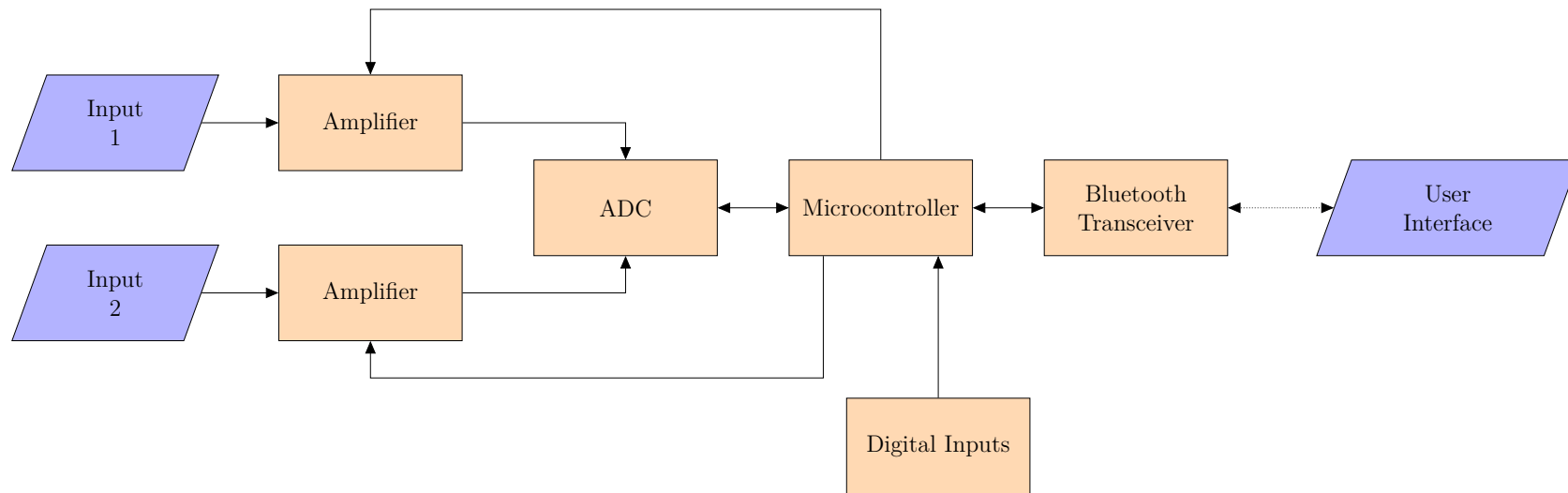


Figure 2.1: Subsystems within the oscilloscope

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 V¹. A non-inverting summing amplifier would not be suitable because 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 will 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 $B = 2.5$ V means that $-V_o$ will vary equally above and below 2.5 V, as we want²(because the voltage must be between 0 V and 5 V, and 2.5 V is the midpoint of those two).

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

²The 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 rheostat (variable resistor), allowing the microcontroller to control the gain of the amplifier.

The input voltage range is from ± 50 mV to ± 5 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 k Ω option will be chosen. This means the potentiometer can vary from the minimum resistance (usually 75 Ω) to 10 k Ω .

Using the values for A and R_1 , this gives a value of $R_f = 5$ k Ω , 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 $R_2 = 10$ k Ω .

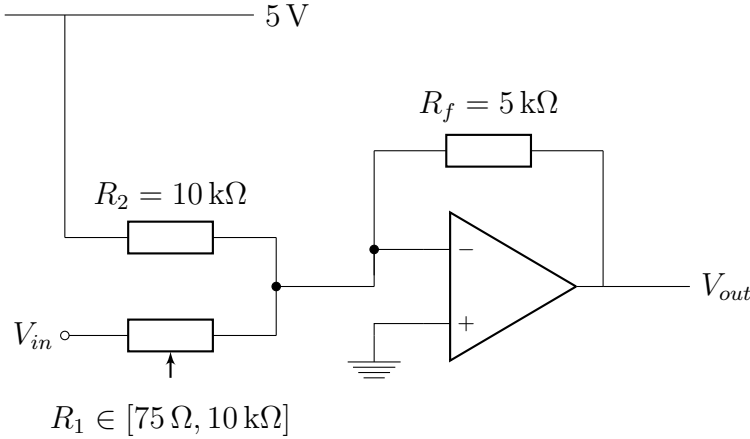


Figure 2.2: First stage amplifier circuit diagram

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 k Ω MCP4131-103 variety will be used.

The op-amps should be of rail-to-rail type (i.e. they can saturate at almost 0 V and V_s) and be able to cope with $V_s = 5$ V, $V_i = 5$ 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 $R_f = 10$ k Ω , we obtain $R_1 = 10$ k Ω .

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

See fig. 2.3 for a circuit diagram.

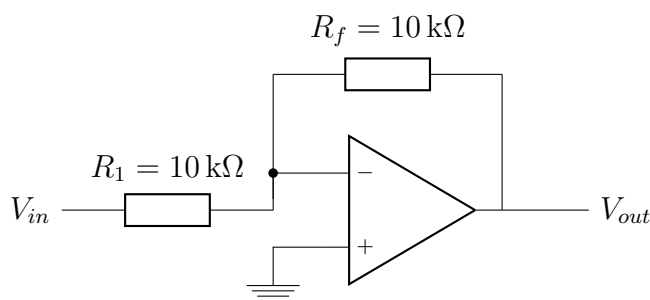


Figure 2.3: Second stage amplifier circuit diagram

Simulation

To check the frequency response of the amplification circuit, it was simulated using LTSpice (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 project³.

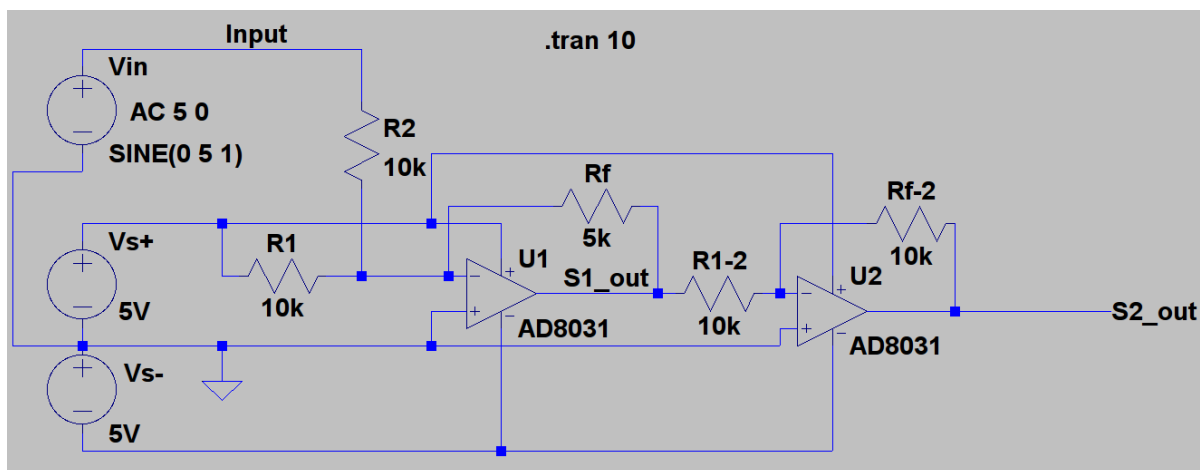


Figure 2.4: Amplification circuit in LTSpice

³The 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

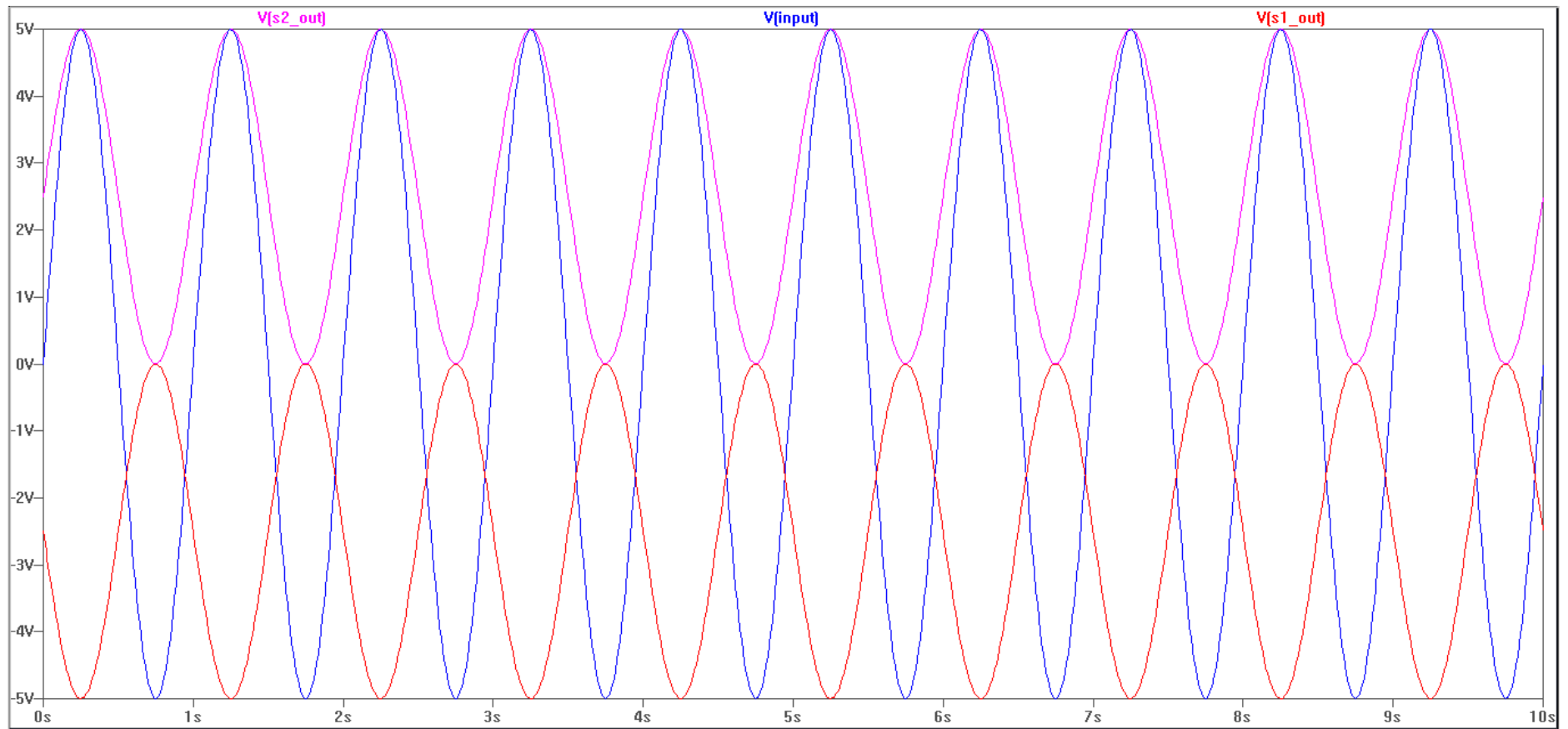


Figure 2.5: Waveform produced by amplification circuit at 1Hz (blue = original, red = first stage output, magenta = second stage output)

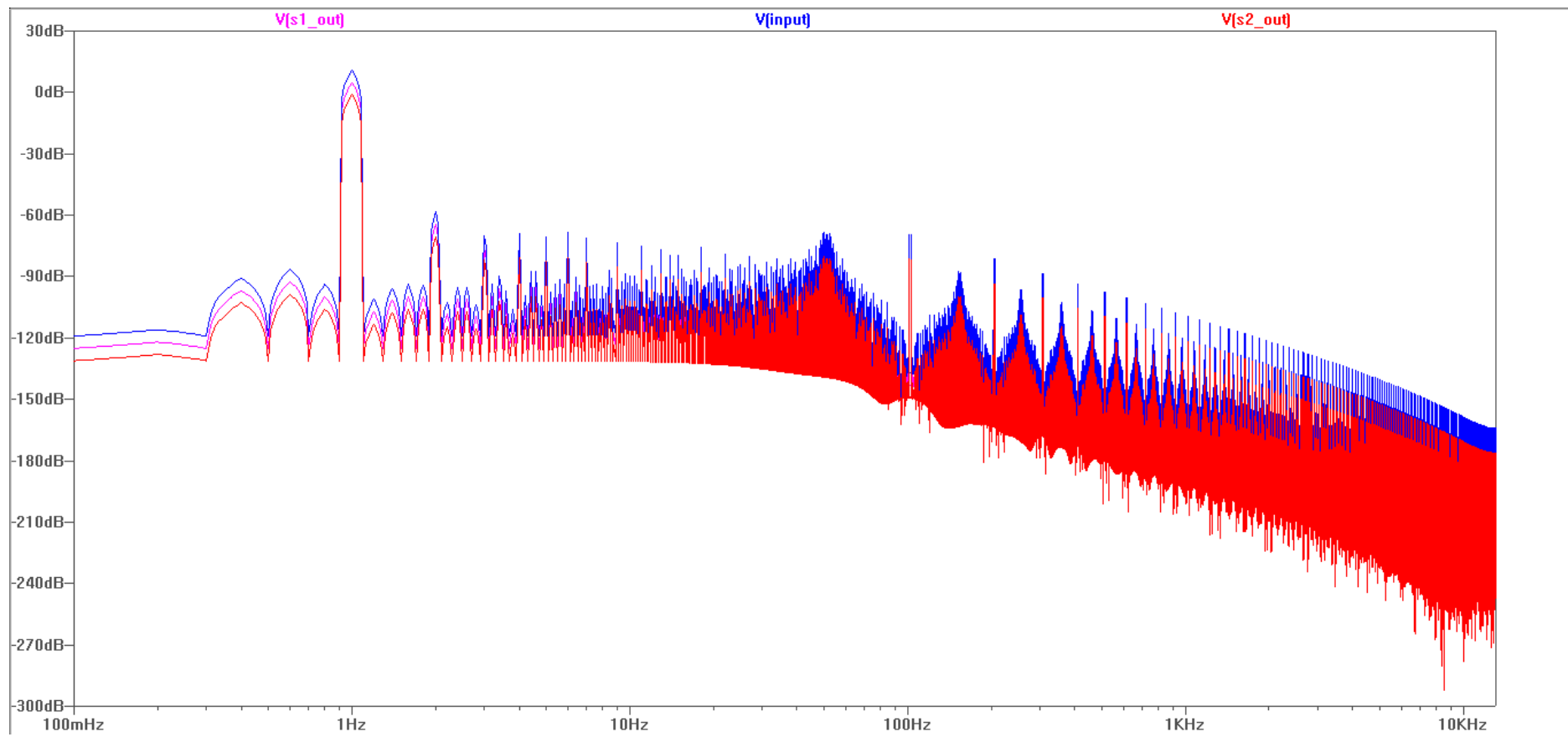


Figure 2.6: FFT of waveform produced by amplification circuit at 1Hz (red = original, blue = first stage output, magenta = second stage output)

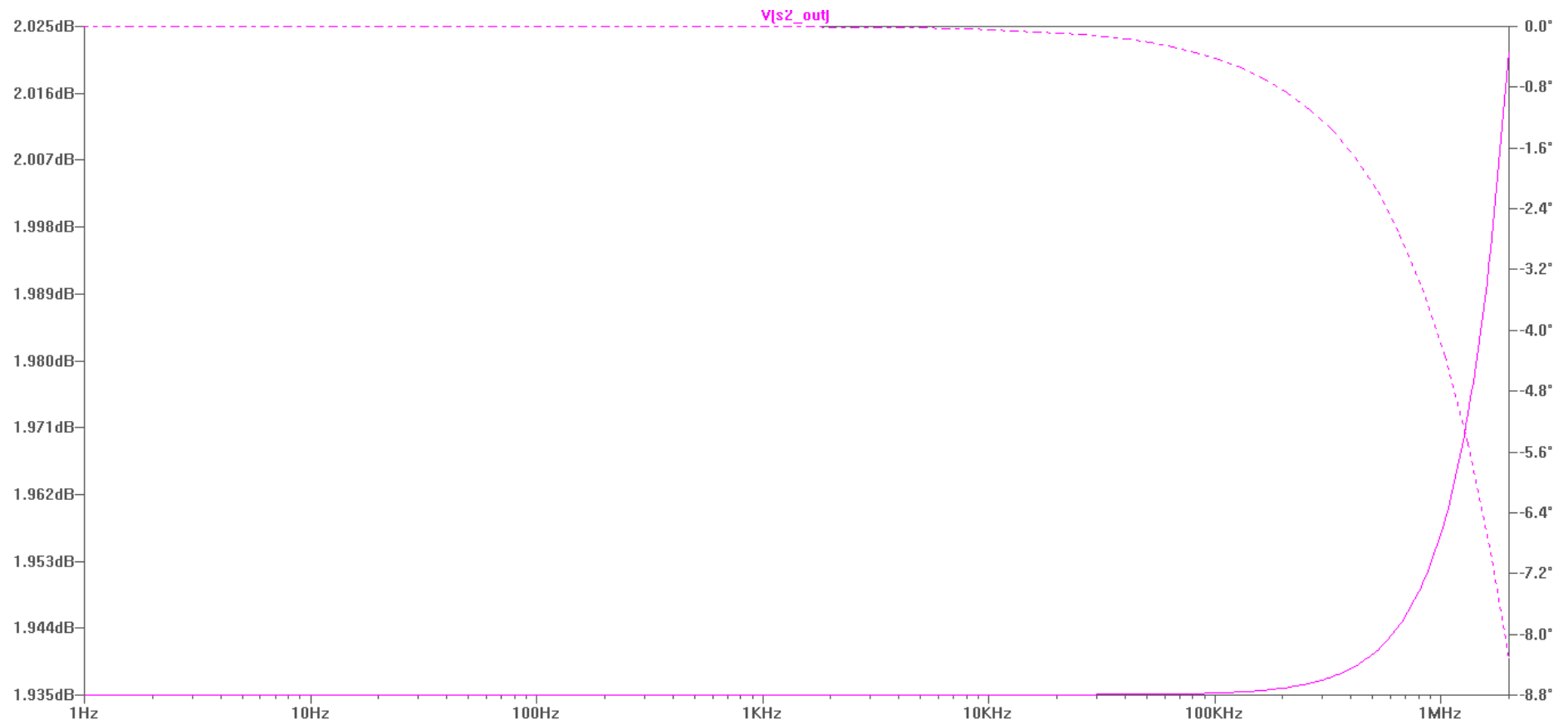


Figure 2.7: Frequency response of amplification circuit

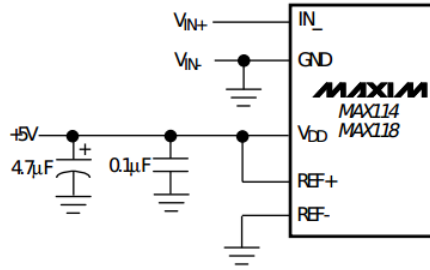


Figure 2.8: Power connections for the MAX114⁶

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-flash⁴ 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

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

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

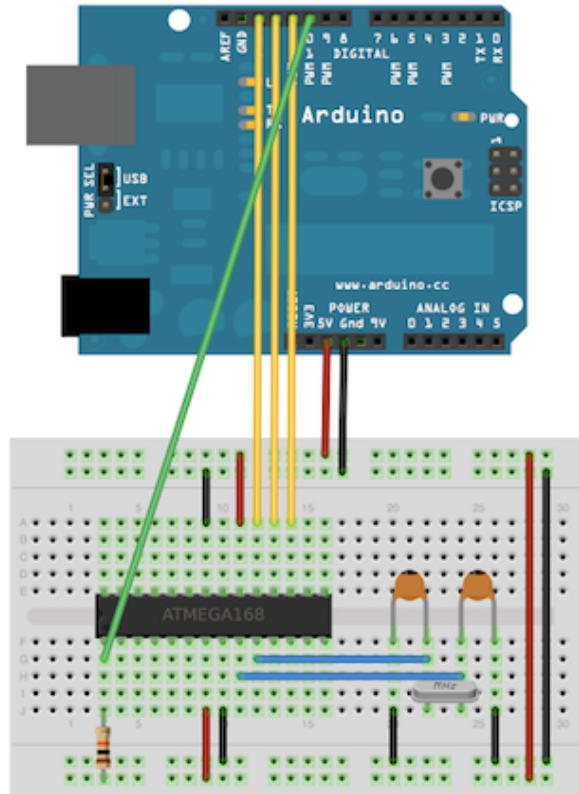


Figure 2.9: Breadboard diagram produced by Arduino (*ArduinoToBreadboard*) showing the programming and power connections between an Arduino and an ATMega328

- 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 k Ω 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 X th 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 X th 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 (returns a 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) (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.

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

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

Digital Potentiometer Communication

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 propagated 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 potentiometers 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 arbitrary 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 oscilloscope. Additionally, it allows easy and open development, whereas competitors such as iOS require costly developer licenses, proprietary software and specific hardware.

2.2 Construction

Initially, a number of the subsystems were built individually on breadboards. This allowed the basic circuit details to be checked without requiring an infeasibly large circuit to be built on a breadboard.

2.2.1 PCB Layout

Once this prototype verification step was performed, the PCB was laid out on a breadboard. This process consists of three main steps: inputting the schematic into the computer, laying out the schematic components onto the board and routing traces (the PCB equivalent of wires) between the components.

Initially this was attempted using a piece of software called Eagle. A significant number of electronics projects use Eagle, from small hobbyist projects right up to industrial products. However, as this author started to lay components onto the PCB using Eagle, one encountered an error: the free version of Eagle only allows a maximum PCB size of 100 mmx80 mm. For reasons discussed later, the PCB size needed to be 100 mmx100 mm. The cheapest paid-version of Eagle is \$169, meaning it was much too expensive to purchase for the sake of this project. Instead, an alternative was sought.

This author researched and briefly evaluated two alternatives: KiCAD and DipTrace. KiCAD is a piece of open source software, meaning it's free to use ⁹, and DipTrace, a

⁹There are two distinct definitions of free that should be considered here: gratis free (without cost to the user, this is somewhat confusingly also called free as in beer) and libre free (without compromise to the user's liberties, this is also known as free as in speech and essentially means that the user has the right to run the software however they want, as well as the right to modify it, meaning the source code is freely available. Software that's Free (with a capital F) is also known as open-source software (there are major differences between the two terms, however for the purposes of this explanation the reader can consider them identical) is both free as in gratis and free as in libre, and common examples include LibreOffice (hence the name) and GNU/Linux (the operating system running everything from the majority of super computers to televisions to traffic lights). For more on this subject, see Richard Stallman's book '*Free as in Freedom*', which is also Free and hence freely available on the internet.

commercial piece of software with a free version available limited to 2 signal layers¹⁰ and 300 pins¹¹.

Both are very capable pieces of software, and KiCAD would certainly be favourable for philosophical reasons. However, the majority of the ICs used in the circuit are relatively uncommon. This means that CAD footprints (files containing the physical dimensions of the chip) are not readily available, so they must be manually created for each IC instead. This is a much quicker process in DipTrace than in KiCAD (and indeed than in Eagle), and for this reason alone DipTrace was chosen.

There were no major issues in creating the circuit schematic in DipTrace, although a number of revisions were required for small improvements. Laying out the PCB and routing the traces was much harder. DipTrace contains tools to automatically place and route, however the electrical engineering community strongly recommends against using auto routers. Placing and routing is a task that's extremely hard for a computer to find an optimal solution to¹². While software can make use of heuristics¹³, it's a commonly held belief that even the best autorouting software pales when compared to a human with a small amount of experience (it may help to think of PCB routing as art: while a computer can be programmed to come up with pieces of art, even a small child can create much better ones).

Because of this, your author set about learning how to place and route PCBs. The American electronics retailer Sparkfun provide a number of online tutorials and videos regarding PCB routing which were extremely helpful. Eventually, the PCB was routed, and while an expert would have done much better it sufficed for the purposes of this project (particularly as there were no high voltages or high-speed signals (≥ 50 MHz) involved). To finish the board, mounting holes and silkscreen¹⁴ (the explanatory text and markings on a PCB) were added.

2.2.2 PCB Production

The final step in the process was to get the board manufactured. In an American or European fab house (the name for the company that prints the boards) this would be extremely expensive. For example, popular EuroCircuits quoted a price of over over over £90 to produce just **one** board.

¹⁰It would be impractical to have all of the traces in a PCB in the same horizontal plane, as they would not be able to cross at any point. Instead, a number of planes, or *layers*, are used. PCBs are made by pressing together a number of 2-sided PCB 'chapters', meaning that common numbers of layers are 2, 4 or 6 (with higher numbers possible but more expensive). As it's the cheapest available and offers enough trace routing, a 2 layer board will be used for this project

¹¹This project uses somewhere in the region of 250 pins, so this limit is not a problem

¹²There is a classification of such problems, known as NP. Very briefly: a *running time* can be assigned to a given algorithm. For example, any algorithm to sort a list of numbers is at most $\mathcal{O}(n \log n)$, meaning that as the list of numbers grows longer, the running time of the algorithm grows slower than $n \log n$, where n is the length of the list. We say an algorithm is polynomial-time if the running time is polynomial (so e.g., it's $\mathcal{O}(n^3)$ but not $\mathcal{O}(2^n)$). We then say an NP problem is one where the solution to the problem can be checked with a polynomial-time algorithm, but there is no polynomial-time algorithm to find that solution. An NP-hard problem is then a problem that is at least as hard as the hardest NP problems. Routing just one trace, with no obstacles to avoid, is an NP-hard problem

¹³A technique designed to find a good-enough solution to a problem, when finding the optimal one would take too long

¹⁴The process is just screen printing, which originally used silk as the screen, hence the name *silkscreen*

Instead, a Chinese proxy to a Chinese fab house was chosen (nearly all low cost Chinese retailers use the Golden Phoenix fab house, but ordering directly from them requires a minimum order of 100 square inches, far beyond the size of this project¹⁵). DFRobot was chosen because it offered more affordable delivery options than the other Chinese options¹⁶. The boards cost \$23.99 for 10 boards, with a \$16 delivery charge on top. At a cost of \$2.40 per board, ordering from a Chinese retailer made the boards over 60 times cheaper than ordering from a European fab house!

As required, the boards were 100 mmx100 mm and 2 layers. There were less choices available (the boards had to be green, and coated in HASL¹⁷) than from a European fab house, but that didn't matter for this project. Due to the recent Chinese New Year, the boards took slightly longer than usual to be produced, although still took only 2 working days, something that EuroCircuits charge over £150 extra for.

2.3 Enclosure

To protect the circuit, an enclosure was needed. There were only a limited number of places where mounting holes could go on the PCB, so a custom-made enclosure had to be made rather than using a prebuilt one with standard mounting hole locations.

The options available to this author were to produce the enclosure using either a 3D printer or a laser cutter. For easy examiner inspection, the enclosure needed to be transparent and only coloured plastic filament was available for the 3D printer, meaning the laser cutter needed to be used.

Rather than using a traditional 3D CAD tool¹⁸, OpenSCAD was used. This is an open source tool that allows 3D objects to be built using *constructive solid geometry*¹⁹. Once the 3D object was created, a small script was written that took horizontal slices of the object at appropriate heights, saving DXFs²⁰ at each height.

¹⁵The proxies do something called panelisation: group a number of small boards onto a larger board, have the large board made at a low cost and then sell on the small boards for a small amount of profit

¹⁶EMS post is universally available for only a couple of dollars, but can take upwards of a month to arrive as it's simply air mail. Many Chinese retailers will also send goods worth hundreds of pounds through EMS post, simply writing '*birthday card*' on the side meaning that while the retailer doesn't have to pay any export fees there is a high chance of the package spending weeks stuck in customs in the UK. Instead, a courier service was chosen. Most Chinese PCB retailers only offer UPS and FedEx, which would have cost around \$30 for this order. DFRobot offered DHL as well, which was just over half the cost at \$16

¹⁷If the boards were to be sold commercially, then HASL couldn't be used as it contains lead so breaks the EU Restriction of Hazardous Substances Directive, but for this hobby project using HASL is fine. This does suggest another reason why European fab houses are more expensive: hazardous substances such as lead can't be used in the board, whereas in China that's not an issue

¹⁸Computer Aided Design tool, such as SolidWorks or AutoDesk Inventor

¹⁹In essence, this allows objects to be built out of primitive objects. For example, a basic hammer could be built out of a cuboid and a cylinder sized and placed appropriately.

²⁰An image format, similar to a JPEG or PNG, that's commonly used with laser cutters

Fourier Series of a Square Wave

Consider a square wave $f(t)$ of amplitude a and period L . An example is shown in fig. A.1.

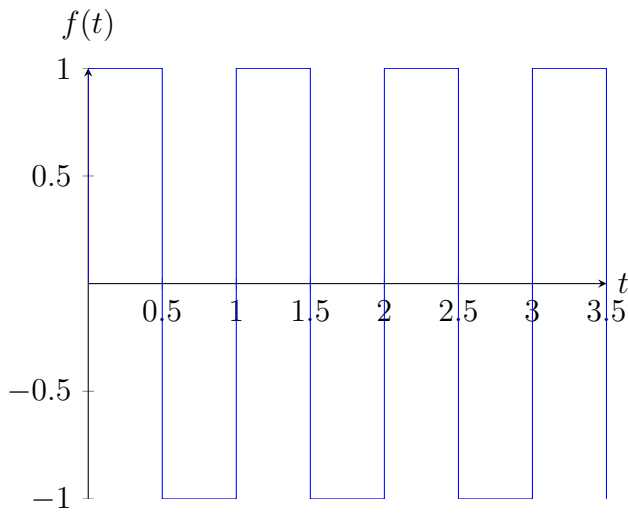


Figure A.1: Square wave with amplitude 1 and frequency 1 Hz

Now by any good mathematical methods textbook (in this case, we cite the excellent Riley, Hobson, and Bence (*Mathematical Methods for Physics and Engineering*)), a periodic function¹ $f(x)$ can be represented by the following series (known as a Fourier series)

$$f(x) = \frac{a_0}{2} + \sum_{r=1}^{\infty} \left[a_r \cos\left(\frac{2\pi r x}{L}\right) + b_r \sin\left(\frac{2\pi r x}{L}\right) \right]$$

where

$$\begin{aligned} a_r &= \frac{2}{L} \int_{x_0}^{x_0+L} f(x) \cos\left(\frac{2\pi r x}{L}\right) dx \\ b_r &= \frac{2}{L} \int_{x_0}^{x_0+L} f(x) \sin\left(\frac{2\pi r x}{L}\right) dx \end{aligned}$$

¹Strictly, the function must meet a number of conditions, called the Dirichlet criteria, but we shall just take it as a given that the square wave does meet all of these (the proof of each of them is essentially trivial depending upon your definition of a square wave)

where x_0 is arbitrary and L is the period of the function.

A handy time-saving trick is to realise that the square wave is an odd function (i.e., $\forall x f(-t) = -f(t)$) so there will be no cosine terms in the series (again, this needs to be made more rigorous, but see a maths textbook) and so $a_r = 0$.

To evaluate b_r , we take $t_0 = -\frac{L}{2}$ and split up the integral into two parts. From $-\frac{L}{2}$ to 0 $f(t) = -a$ and from 0 to $\frac{L}{2}$ $f(t) = +a$. We then have

$$\int_{-\frac{L}{2}}^{\frac{L}{2}} f(t) \sin\left(\frac{2\pi r t}{L}\right) dt = a \int_{-\frac{L}{2}}^0 -\sin\left(\frac{2\pi r t}{L}\right) dt + a \int_0^{\frac{L}{2}} \sin\left(\frac{2\pi r t}{L}\right) dt$$

which, using the substitution $u = -t$ on the left integral, equals

$$2a \int_0^{\frac{L}{2}} \sin\left(\frac{2\pi r t}{L}\right) dt = \frac{aL}{\pi r} \left[\cos\left(\frac{2\pi r t}{L}\right) \right]_{\frac{L}{2}}^0 = \frac{aL}{\pi r} [1 - \cos(\pi r)]$$

Noting that $\cos 0\pi = 1$, $\cos 1\pi = -1$, $\cos 2\pi = 1$ and so on, we obtain

$$b_r = \frac{2a}{\pi r} [1 - (-1)^r] = \begin{cases} r \text{ even} & 0 \\ r \text{ odd} & \frac{4a}{\pi r} \end{cases}$$

and hence

$$\boxed{f(t) = \frac{4a}{\pi} \left(\sin \omega t + \frac{\sin 3\omega t}{3} + \frac{\sin 5\omega t}{5} + \dots \right)}$$

where the angular frequency $\omega = \frac{2\pi}{L}$. Plotting the first terms of this equation gives the various graphs shown in fig. 1.6.

List of Corrections

More on digital pot. communication	30
--	----

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] jepoirrier. *Oscilloscope with noise*. accessed 18/02/14. URL: <http://flic.kr/p/6uyTN1>.
- [6] Maxim Integrated. *MAX114/MAX118 Datasheet*. accessed 02/01/14. URL: <http://datasheets.maximintegrated.com/en/ds/MAX114-MAX118.pdf>.
- [7] mightyohm. *New Oscilloscope*. accessed 18/02/14. URL: <http://flic.kr/p/6rdh6K>.
- [8] mik3y. *usb-serial-for-android*. accessed 26/01/14. URL: <https://github.com/mik3y/usb-serial-for-android>.
- [9] W. Phillips. *Using an Oscilloscope*. accessed 26/01/14. URL: <http://www.doctrionics.co.uk/scope.htm>.
- [10] Picotech. *Advanced Triggering with PicoScope*. accessed 02/01/14. 2014. URL: <http://www.picotech.com/education/oscilloscopes/advanced-triggering.html>.
- [11] K.F. Riley, M.P. Hobson, and S.J. Bence. *Mathematical Methods for Physics and Engineering*. 2nd ed. Cambridge University Publishing, 2002.
- [12] 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>.
- [13] Semtech. *Improving the Accuracy of a Crystal Oscillator*. accessed 18/02/14. URL: www.semtech.com/images/datasheet/xo_precision_std.pdf.
- [14] Claude Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27 (July 1948), pp. 379–423, 623–656. URL: <http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf>.
- [15] Silicon Labs. *Single-chip USB to UART Bridge*. accessed 26/01/14. URL: <http://www.silabs.com/Support%20Documents/TechnicalDocs/CP2102-9.pdf>.
- [16] Deepak Kumar Tala. *ASIC World*. accessed 01/02/14. URL: <http://www.asic-world.com/verilog/index.html>.
- [17] TiePie Engineering. *Digital Data Acquisition*. accessed 26/01/14. URL: http://www.tiepie.com/en/classroom/Measurement_basics/Digital_Data_Acquisition.

- [18] Alan Tong. *What to Look for When Choosing an Oscilloscope*. accessed 02/01/14. 2014. URL: http://www.picotech.com/applications/oscilloscope_tutorial.html.
- [19] Wikipedia. *Verilog — Wikipedia, The Free Encyclopedia*. accessed 01/02/14. 2013. URL: <http://en.wikipedia.org/wiki/Verilog>.
- [20] Joel Woodward. *What's the Difference Between 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>.