Physics 4050H: Project 1

Jeremy Favro (0805980), Melody Berhane ()
Department of Physics & Astronomy
Trent University, Peterborough, ON, Canada

October 3, 2025

Todo list

Abstract

We successfully automated PHYS-2250H's lab 4 on operational amplifiers through microcontroller operated wave generation and input element switching, however due to time constraints did not automate data collection. The automation functions well with issues only in the imperfect connections of the breadboard sometimes causing disconnects when disturbed.

1 Introduction

The goal of this project was to take a lab from PHYS-2250H (Electronics) and improve upon/automate the content of that lab. We opted to automate lab 4 which involved constructing different configurations of operational amplifiers and verifying that the theoretical equations derived in lecture corresponded to the real-world behaviour of the device. Lab 4 specifically looked at varying the input resistance of an inverting amplifier and looked at a single configuration of differentiating amplifier. Our initial concept for automation was to go the most direct route and use a microcontroller to control a switch which would allow selecting different components as in the case of both the inverting and differentiating amplifiers the only component which is varied between measurements is the element lying between the signal source and inverting input of the amplifier. We additionally sought to eliminate as many components external to those which sit directly on a breadboard and so created a (single) function generator to replace the one used in the original lab.

2 High Level Overview

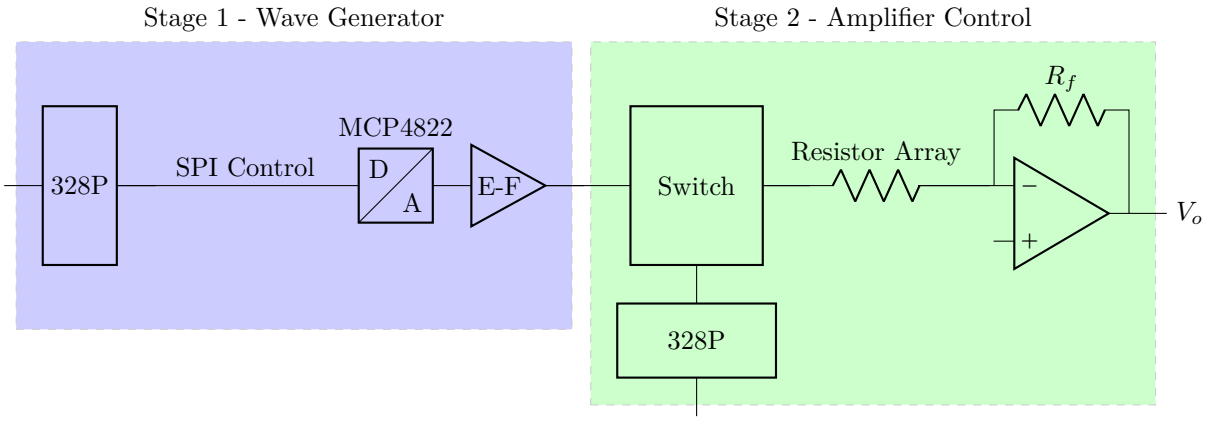


Figure 1: Block Diagram of the Device. Note that the component labeled E-F is an emitter-follower/voltage follower op-amp

The automation consists of two main stages, one which generates a fixed frequency and peak-to-peak voltage input and one which controls which element is connected to the inverting input of the amplifier. There is technically a third stage which provides power to the devices in the other two but it is a basic three component circuit comprised of a 5 V regulator and two filter capacitors and is therefore left off of diagrams from hereon.

2.1 Wave Generator

The input signal generator uses a single ATMmega328P [2] (hereon referred to as a 328P) microcontroller connected through SPI to an MCP4822[1] digital-to-analog converter (DAC from hereon). The output of the DAC is passed through a low-pass reconstruction filter, then to a emitter-follower/voltage-follower op-amp to stabilize the signal and then forwarded to the second stage of the device.

2.2 Amplifier Control

This section of the device uses another 328P to control a 4 input/output switch where each input is connected to the output of the first stage (the sine wave). Each output is connected then to a single resistor each of which are in turn connected to the inverting input of an op-amp wired in an inverting configuration. The 328P switches between each of these resistors every two seconds.

3 Detailed Methods

3.1 Wave Generator

The 328P runs the code seen in 6.2. The "algorithm" used in the code is known as a phase accumulator. A phase accumulator is convenient as it allows easy control of the frequency of the output wave desired frequency and the number of samples in the lookup table. At the beginning of the program we calculate a period over which the phase accumulator operates and populate a lookup table with precalculated values of our desired output wave, sin in this case. Then, for every iteration of the loop, we first check if sufficient time has passed to update the output of the DAC. If not, we continue the loop. If sufficient time has passed we first write out the value in our lookup table which corresponds to the current "phase value". We then increment the phase modulo the size of our table so that it will wrap around to the beginning when we reach the end of the table, then increment the next update time by our precalculated period. Writing a value to the DAC involves, for the MCP4822, sending a 16-bit integer via SPI. The contents of this integer, are, ordered from most to least significant bits, 4 command bits whose meanings are shown in 6.2 for our specific case and explained more generally in the datasheet [1], then the 12-bit value to be written to the output of the DAC. Dependent on command bits, this value may be a fraction of either 4096 (twice the internal voltage reference of the MCP4822) or 2048. As our application requires a 1V peak-to-peak wave, we opted for the latter. The 12-bit value is then represented at the output of the DAC as the given fraction of the internal voltage

reference. As it becomes difficult to write with sufficient speed to produce a high-fidelity wave directly from the DAC when high-frequencies are desired we only operate with 16 possible states for the output of the DAC. Because we desire a smooth wave for the input of the op-amp we need to feed the stepped output that this low sample count results in through a suitable reconstruction filter. Here this will be a filter which rejects the sudden changes of the stepped wave and makes them more gradual, e.g. a low-pass filter. The output of the DAC is already fed through a $1\,\mathrm{k}\Omega$ current limiting resistor so we opted to include a $1\,\mu\mathrm{F}$ capacitor to ground in parallel with this output resistor, giving a time constant of $1\,\mathrm{ms}$ which, though not exactly the same as our sampling frequency, gives a sufficiently smoothed wave and was chosen based on on-hand components. For the final step of this first stage of the whole device we feed the signal through an emitter-follower amplifier to pass the loading created by the second stage from the DAC to an op-amp which fixes the output voltage at the voltage generated by the DAC.

3.2 Amplifier Control

The amplifier control circuit is much simpler than the frequency generator and is comprised of a single 328P running the code found in 6.1 which is dedicated to switching four of its digital output pins between high and low in sequence. These pins are connected to the control pins of an SN74HC4066N [3] whose input pins are chained together and connected to the output of stage 1 (the emitter-follower output). Each output pin of the switch is then connected to a $220\,\Omega, 1\,\mathrm{k}\Omega, 3.6\,\mathrm{k}\Omega, 9.1\,\mathrm{k}\Omega$ resistor, each of which are in turn connected to the inverting input of our final op-amp. We then have the inverting input connected to the output through a $10\,\mathrm{k}\Omega$ resistor (and the non-inverting input connected to ground) to create a variable inverting amplifier.

4 Conclusion

Generally the project was a success insofar as automation is concerned.

4.1 Areas for Improvement

4.1.1 Automated Measurement

The 328P is capable of analog reads from $0 - 5\,\mathrm{V}$. The amplifier stage of the device outputs above this range but could be divided down with a voltage divider to allow automatic measurement of both the input and output signals of the inverting amplifier. Depending on desired functionality these signals could be sent over USB (though this would require a suitable interface if the 328P was to remain in the breadboard rather than the Arduino) or to a breadboard mounted display to eliminate the need for external measurement devices.

4.1.2 Current Element Display & Manual Element Switching

Currently the arduino controlling the switch operates on a two second timer where each input element is on for two seconds before the device switches to the next. As an extra feature, it would not be difficult to add a button or other input device which would allow a user to decide when they want to switch between different input elements. In addition to this it may be nice to have a display controlled by the switching 328P to indicate which input element is currently selected. This could be a display element such as an LCD or just LEDs tied to the control inputs of the switch to indicate which is selected, though an LCD would be more clear.

4.1.3 Differentiating op-amp

The original version of the lab automated here included a fifth input element, a $0.1\,\mu\mathrm{F}$ capacitor in the same location as the resistors to create a differentiating amplifier. Adding this alongside the 4 resistors would require either a different switch with more I/O pins or a second switch. One of the existing resistors could also easily be switched out for a capacitor, probably either of the two middle valued ones as the upper and lower ones exhibit more interesting properties with $220\,\mathrm{k}\Omega$ having a clipped waveform due to over driving the amplifier and $9.1\,\mathrm{k}\Omega$ having a gain of near 1 and therefore acting nearly as an inverter alone.

5 References

References

[1] Microchip Technology. *8/10/12-Bit Dual Voltage Output Digital-to-Analog Converter with Internal VREF and SPI Interface*, 4 2015. Rev. B.

[2] Microchip Technology. *ATmega48A/PA/88A/PA/168A/PA/328/P*, 9 2020. Rev. B.

[3] Texas Instruments. *Quadruple Bilateral Analog Switch*, 2 2024. Rev. K.

6 Appendix

6.1 Switch Control Code

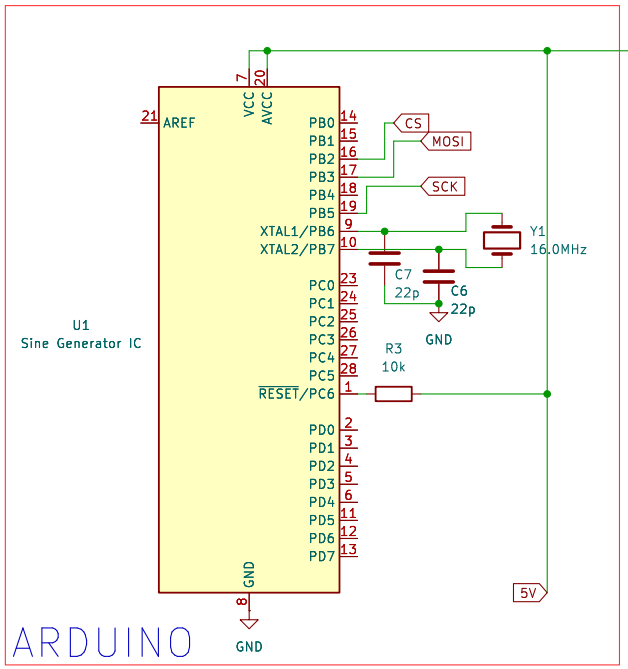
```
const uint32_t SW_PERIOD_US = 2e3;  // 2s
int cur_pin = 2;

void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  Serial.begin(9600);
}

void loop() {
    digitalWrite(cur_pin, LOW);

    cur_pin++;
    if (cur_pin > 5) {
      cur_pin = 2;
    }
    switch (cur_pin) {
      case 2:
        Serial.print("220Ω");
        Serial.print("\n");
        break;
      case 3:
        Serial.print("1kΩ");
        Serial.print("\n");
        break;
      case 4:
        Serial.print("3.6kΩ");
        Serial.print("\n");
        break;
      case 5:
        Serial.print("9.1kΩ");
        Serial.print("\n");
        break;
      default:
        break;
    }
    digitalWrite(cur_pin, HIGH);
    delay(SW_PERIOD_US);
}
```

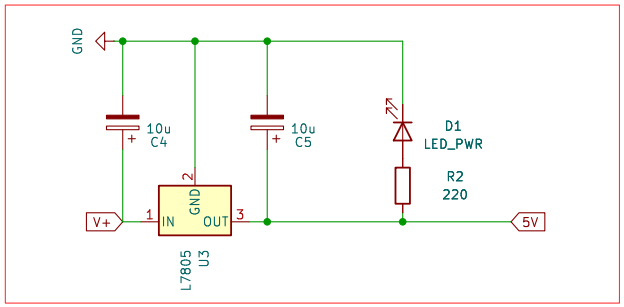
6.2 Frequency Generator Code

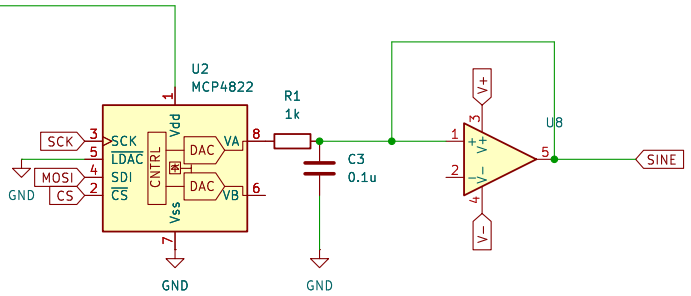
```
#include <SPI.h>
#define SAMPLES 16
#define FREQUENCY 1000
#define CS_PIN 10
const uint32_t PERIOD = 1e6 / (FREQUENCY * SAMPLES);
uint32_t next = 0;

uint16_t phase = 0;

uint16_t SIN_LUT[SAMPLES] = {};

void setup()
{
  // Populate lookup table @ ~1V_pp
  for (int k = 0; k < SAMPLES; k++)
  {
    SIN_LUT[k] = ((2048 / 2) * (sin(2 * PI * (float)k / SAMPLES) + 1));
  }

  SPI.begin();

  // Chip select pin active low, this will deselect the DAC
  digitalWrite(CS_PIN, HIGH);
}

void loop()
{
  uint32_t now = micros();

  if (now > next)
  {
    writeDAC(SIN_LUT[phase]);

    phase = (phase + 1) % SAMPLES;
    next += PERIOD;
  }
  // Could we delayMicroseconds(PERIOD);??
}

void writeDAC(uint16_t value)
{
  uint16_t command = 0x3000; // 0011...,
  // 0 -> Write Channel A
  // 0 -> DC
  // 1 -> V_o=V_r * value/4096
  // 1 -> Output shutdown, operating on HI
  // See datasheet for more

  // AND here forces to 12 bit
  // OR command w/value, gives
  // 0011 value
  command |= (value & 0x0FFF);

  digitalWrite(CS_PIN, LOW);

  SPI.transfer16(command);

```

```
  digitalWrite(CS_PIN, HIGH);
}
```

6.3 Full Schematic



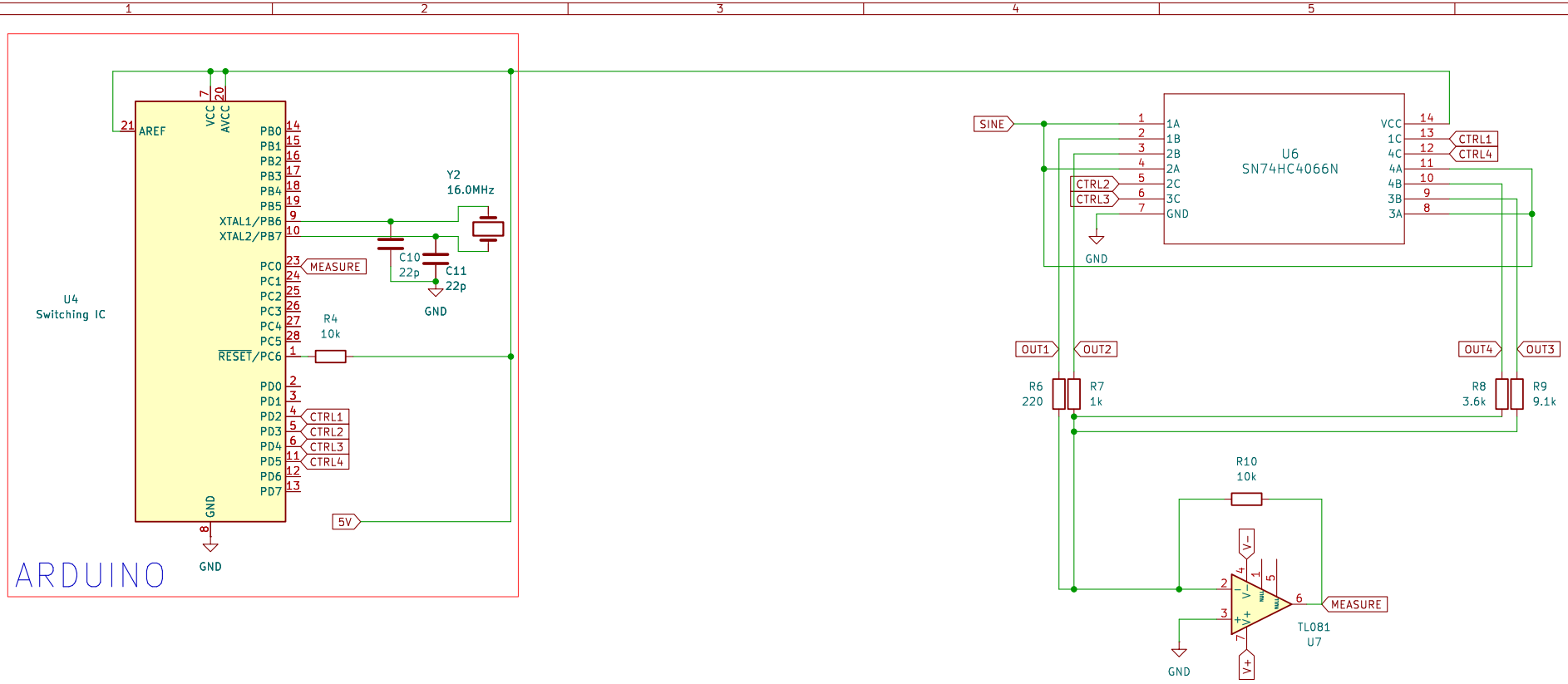




Measurement & Switching

File: MeasurementSwitching.kicad_sch

Sheet: /		
File: Kicad Schematics.kicad_sch		
Title: Sine Generator		
Size: A4	Date: 2025-09-12	**Rev: 1**
KiCad E.D.A. 9.0.4		Id: 1/2



Sheet: /Measurement & Switching/		
File: MeasurementSwitching.kicad_sch		
Title: Measurement & Switching		
Size: A4	Date: 2025-09-13	**Rev: 1**
KiCad E.D.A. 9.0.4		Id: 2/2