Simple Circuit Projects is about making circuits that are educational.

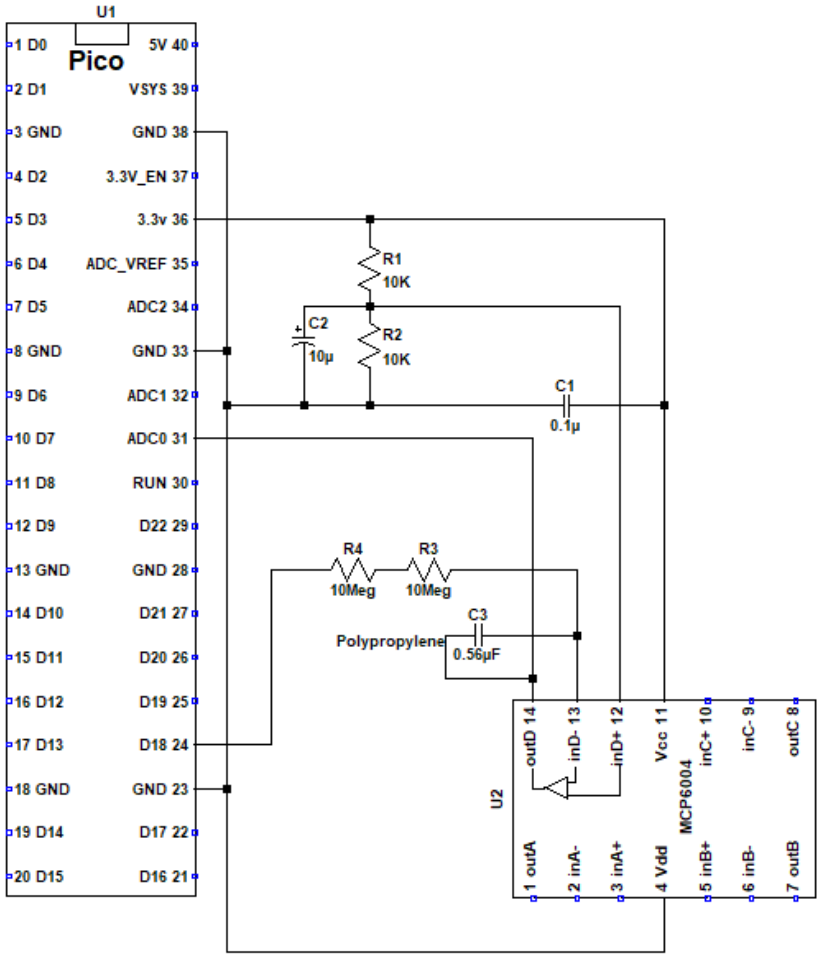
This pdf is a step-by-step process for measuring the Raspberry Pi Pico ADC linearity. The method is applicable for other processors with analog to digital converters.

This project requires:

- A text editor
- A spread sheet
- The Arduino IDE
- Earle Philhower's arduino-pico must be installed.
    - Go to the Github web page and follow the instructions.
    - https://github.com/earlephilhower/arduino-pico

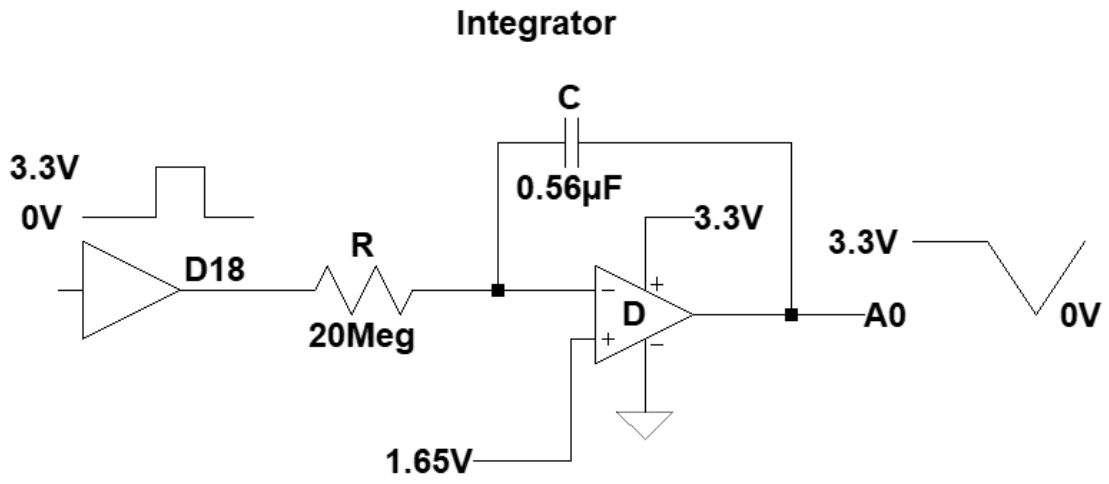
In addition to the Pico, you will need:

- A rail-to-rail CMOS operational amplifier.
    - We use one of the amps on a MCP6004.
- A solderless breadboard.
- Four resistors.
- Three capacitors fill out the component count.
- You will also need some jumper wires.
    - Homemade ones from 22 gage solid wire are ideal as they are cut to length.



The circuit uses two 10K resistors to divide the 3.3V supply to 1.65V. A 10uF electrolytic capacitor filters noise from the 3.3V supply. 1.65V connects to the non-inverting input and is used as the operational amplifier ground reference.

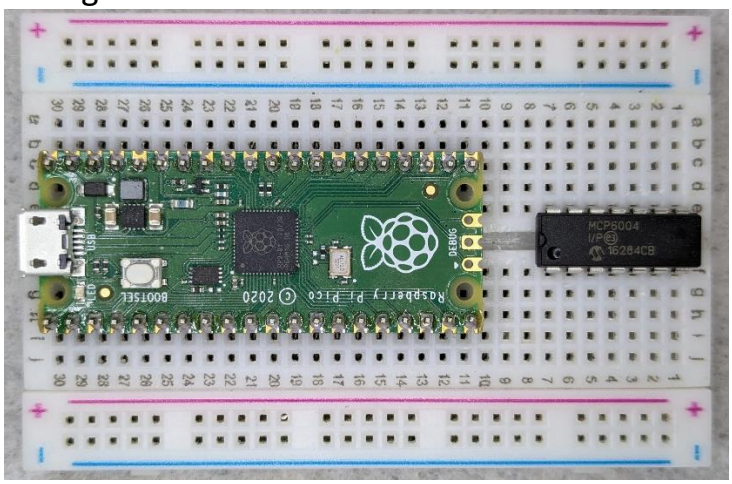
Two 10meg resistors are connected in series between logic output D18 and the inverting input. A 0.56uF polypropylene capacitor connects from the inverting input to the output.



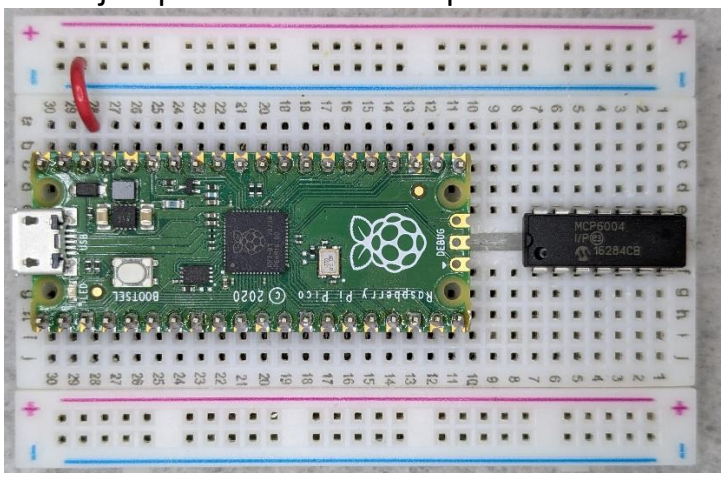
The op amp is configured as an inverting integrator with a gain of 1/RC or 1/11.2seconds. When D18 is low or zero volts, the input is -1.65V relative to the non-inverting input. The output will linearly increase in voltage at a rate of 1.65V/11.2sec = 0.147V/sec. It takes 22.4 seconds to ramp from zero to 3.3V.

When D18 is high or 3.3V, the integrator ramps towards ground at an identical rate. The linear ramp has a near perfect slope and it can be used to compare the linearity of the ADC. We just need to sample the ramp at very consistent intervals.

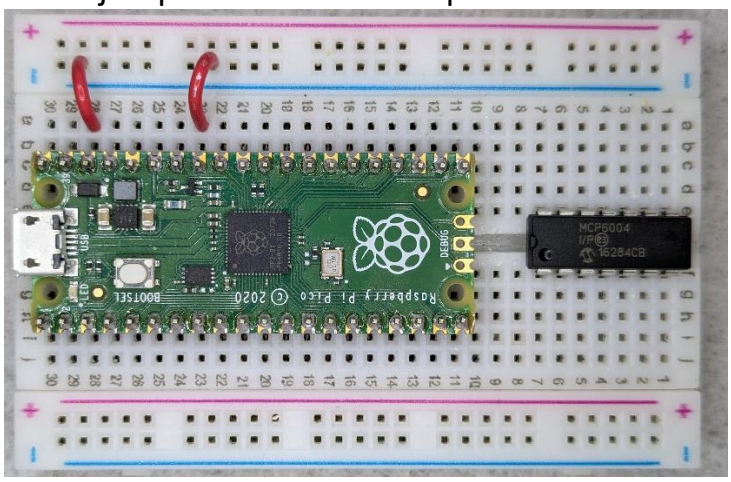
Let’s get started. Insert the Pico and MCP6004 into the bread board.



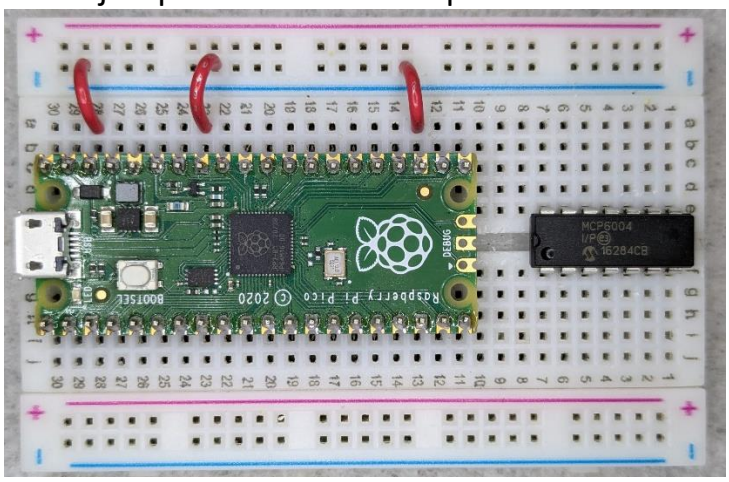
Add a jumper wire from Pico pin-38 to the minus bus.



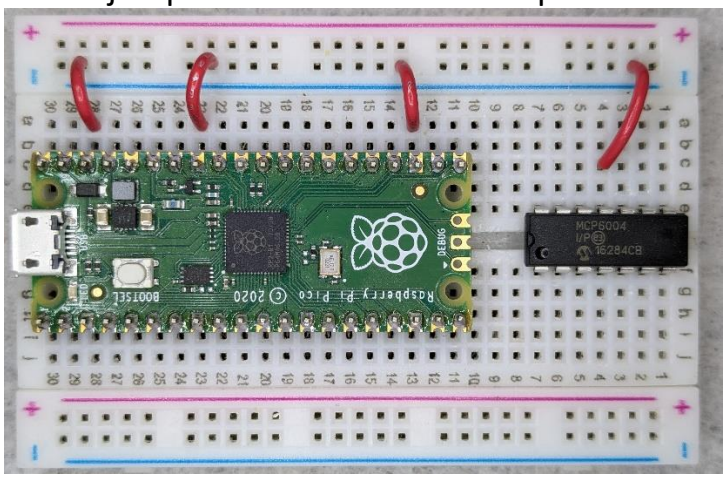
Add a jumper wire from Pico pin-33 to the minus bus.



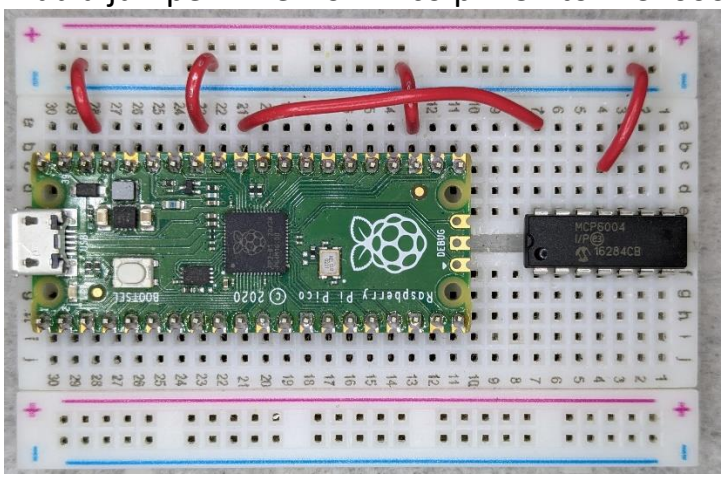
Add a jumper wire from Pico pin-23 to the minus bus.



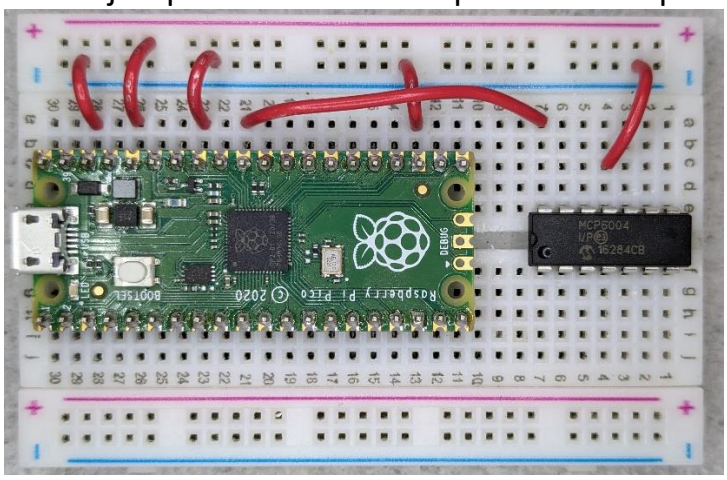
Add a jumper wire from MCP6004 pin-11 to the minus bus.



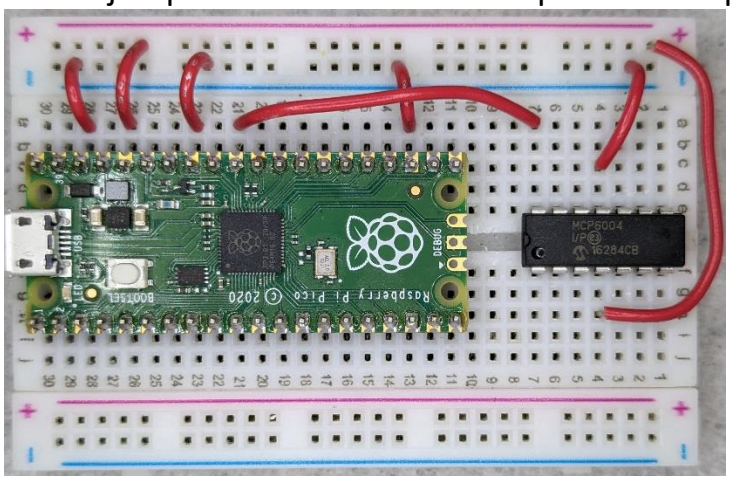
Add a jumper wire from Pico pin-31 to MCP6004 pin-11.



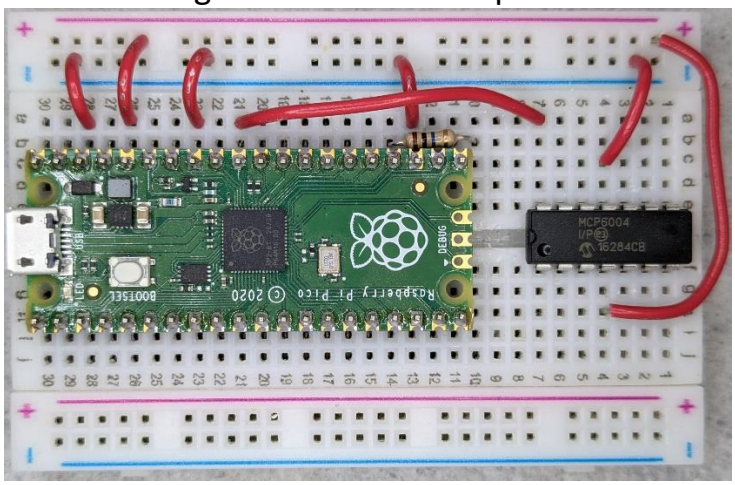
Add a jumper wire from Pico pin-36 to the plus bus.



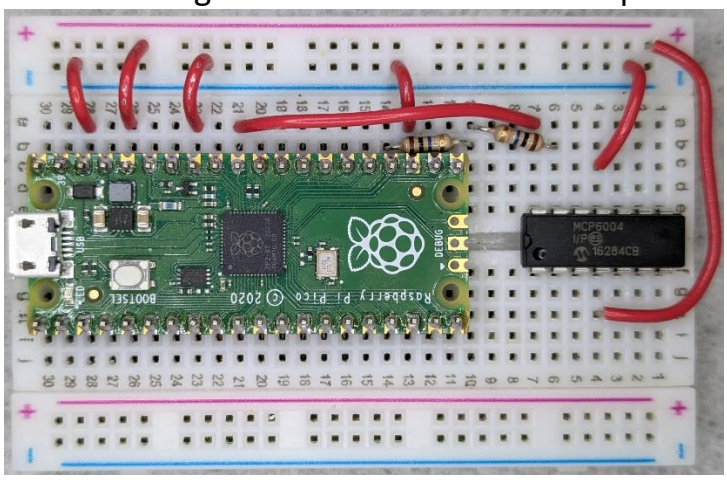
Add a jumper wire from MCP6004 pin-4 to the plus bus.



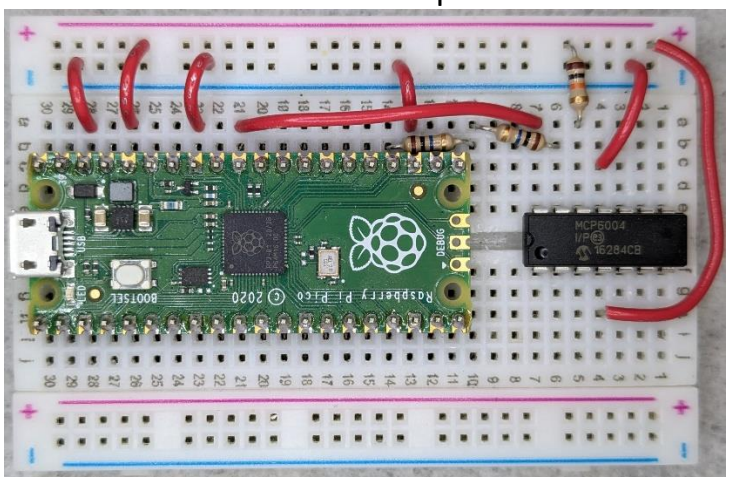
Add a 10meg resistor from Pico pin-24 to a vacant strip.



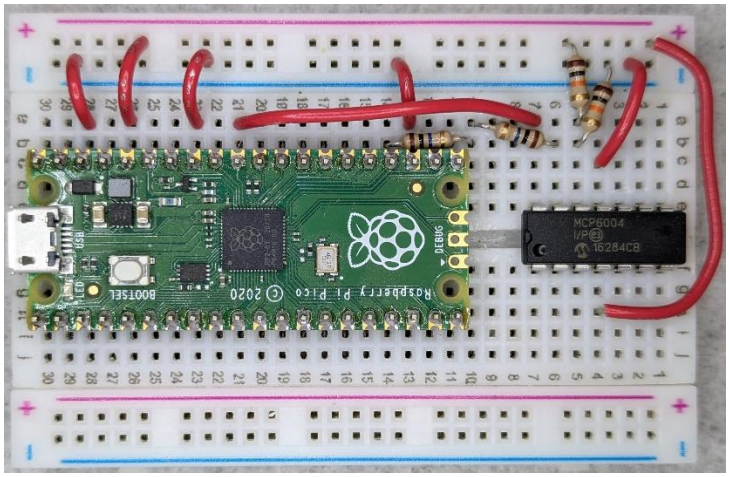
Add a 10meg resistor from the same strip to MCP6004 pin-13.



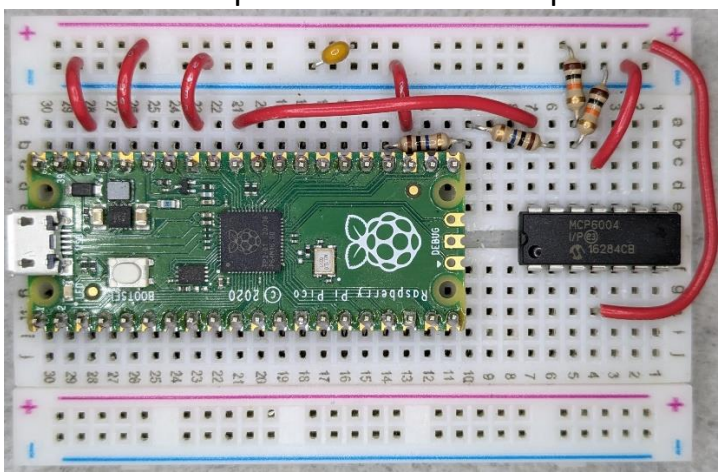
Add a 10K resistor from the plus bus to MCP6004 pin-12.



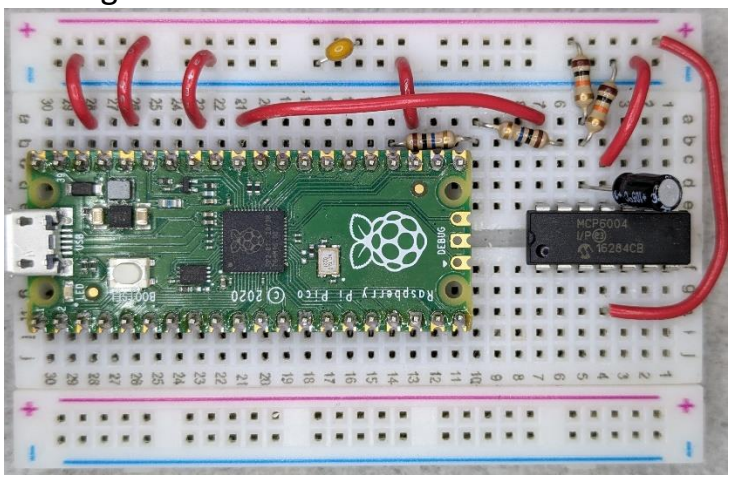
Add a 10K resistor from the minus bus to MCP6004 pin-12.



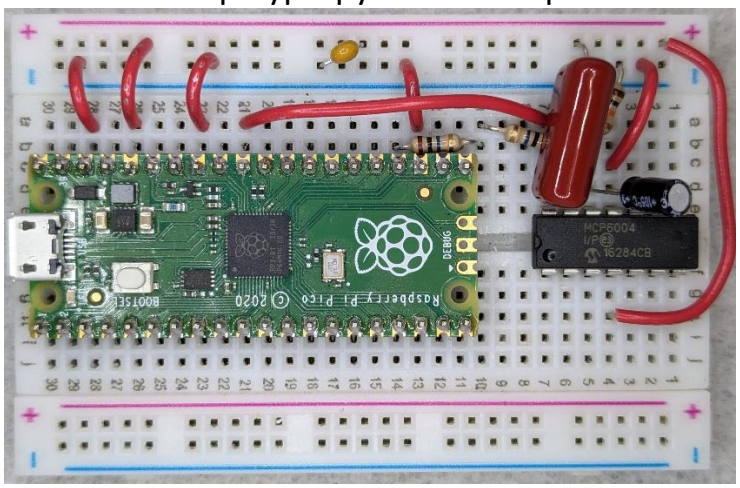
Add a 0.1uF capacitor between the plus and minus bus.



Add a 10uF electrolytic capacitor between MCP6004 pins 11 and 12. Pin 11 is the negative terminal.



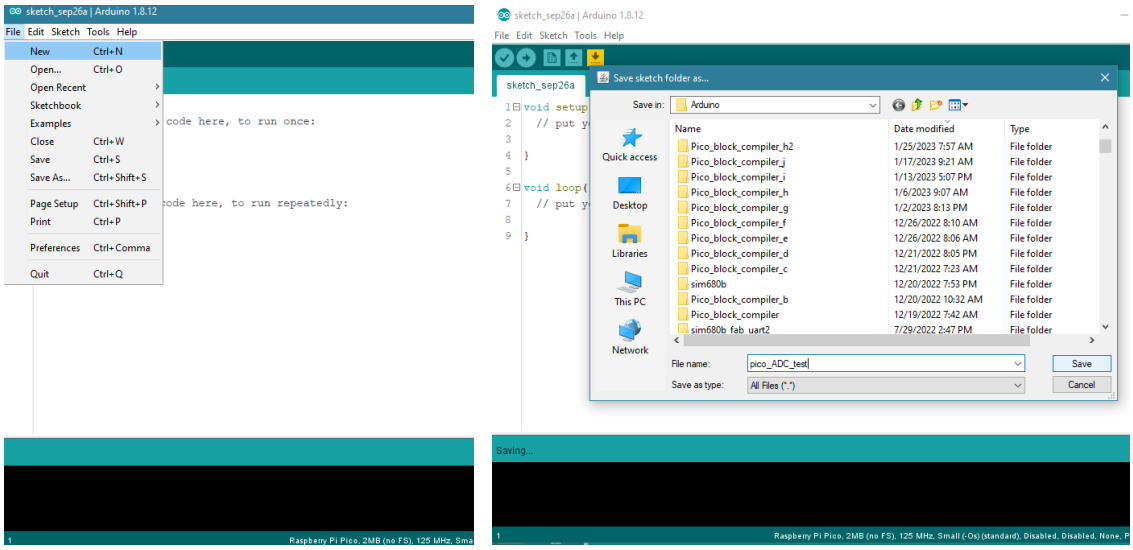
Add a 0.56uF polypropylene film capacitor between MCP6004 pins 13 and 14.



The test algorithm only takes a few steps:

- The integrator is preset for a positive ramp
- After lower case ‘r’ character is received, the test starts
- Text output indicates the integrator is ramping negative towards ground. D18 is set high.
- The output of the integrator comes close to ground but will never reach zero. To ensure the ADC reads a value, a threshold above zero is used.
- A delay ensures the output swings past the threshold
- Text output indicates the integrator is ramping positive towards 3.3V. D18 is set low.
- Once the ADC detects a small positive threshold, the ADC value is sent every 5ms.
- Sampling stops when a threshold just under the maximum 12-bit 4095 value is received.

Start the Arduino IDE select the File drop down and click on new. Save the sketch as linearity_tester_pico



Let's start adding code.

Two global variables are assigned. One is for ADC averaging and the other for timing the sample intervals.

```
1  //ADC differential linearity test using integrator ramping. Simple-Circuit 2023
2  volatile long analog_avg;
3  volatile unsigned long t;
4
```

Note: The program was compiled using arduino-pico version 1.1.0. The latest version 3.6.0 requires the ADC to be set to 12-bits resolution. Add the following code to line 4:

analogReadResolution(12);

The system is configured for D18 as an output and serial communication.

```
5  void setup() {
6   pinMode(18,OUTPUT);
7   digitalWrite(18,0);
8   Serial.begin(460800);        //set your serial terminal to 460800 baud
9  }
10
```

Eight ADC readings are averaged to reduce noise. The eight reads take about 80us to complete. One bit change is 3.3V/4096 or 0.8mV. The integrator changes 50us*0.147V/sec which equals 0.008mV. This is much less than one LSB.

```
11 void readA0(void){
12       analog_avg = 0;
13       for (int j = 0; j<8; j++) analog_avg += analogRead(A0);  //average 8 readings to reduce noise
14       analog_avg =  analog_avg >> 3;                           //shift to 12-bit value
15 }
16
```

D18 is set to ramp the integrator to 3.3V while we wait for a small r character.
D18 is set to ramp the integrator to zero.

```
void loop() {
  char c;
   c=' ';
   digitalWrite(18,0);
   while (c!='r'){                    //start the integrator test when an 'r' is typed
     while (Serial.available() == 0);
     c = Serial.read();
   }
```

A start text is sent. A loop waits for the ADC to reach a count of 30.

```
     Serial.println("Ramping Down");  //Set DAC1 to the mid point
     digitalWrite(18,1);              //is set to max positive voltage
     readA0();
     while (analog_avg > 30){        //wait for the integrator to reach zero
      readA0();
     // Serial.println(analog_avg);
     }
```

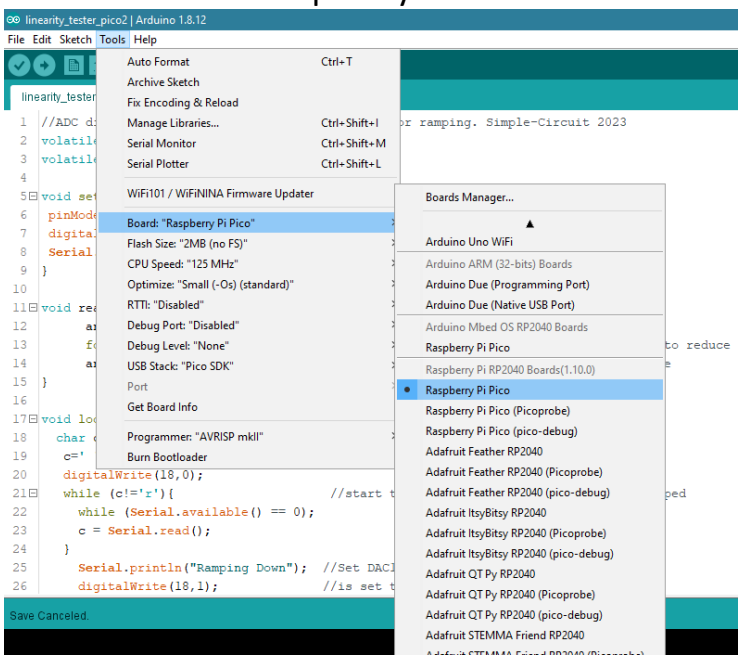
A start text is sent. After a short delay, D18 is set to ramp the integrator to 3.3V.
The ADC is sampled until the ADC reaches a count of 25

```
     Serial.println("Ramping Up");
     delay(100);                     //delay to let integrator go slightly negative
     digitalWrite(18,0);             //set DAC0 below DAC1 volage to ramp positive
     readA0();
     while (analog_avg < 25){       //noise can cause false starts, so wait for
      readA0();                      //a higher value
     }
```

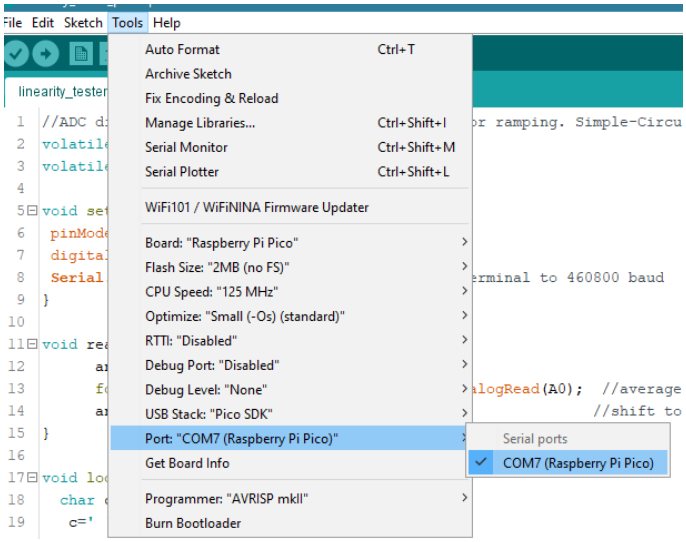
The ADC is read every 5ms and the value sent. Reading continues until the integrator reaches a count of 4070 which is 25 less than the max of 4095.

```
     t = millis()+5;                 //set to sample at 5ms intevals
     while (analog_avg < 4070){     //print ADC values intil max is reached
      readA0();
      Serial.println(analog_avg);
      while (millis() < t);
      t = t + 5;
     }
}
```

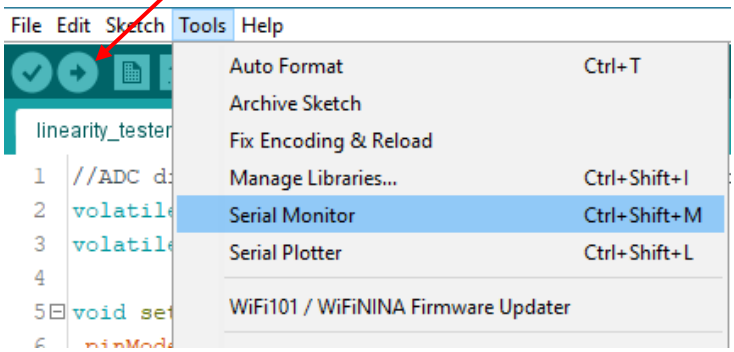
To program the Pico, connect your Pico to a USB port. In the Tools menu, set the Processor to Raspberry Pi Pico.



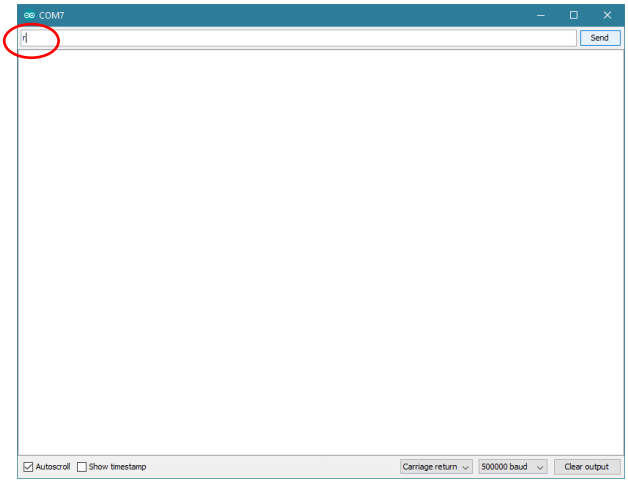
Make sure to select the active serial port.



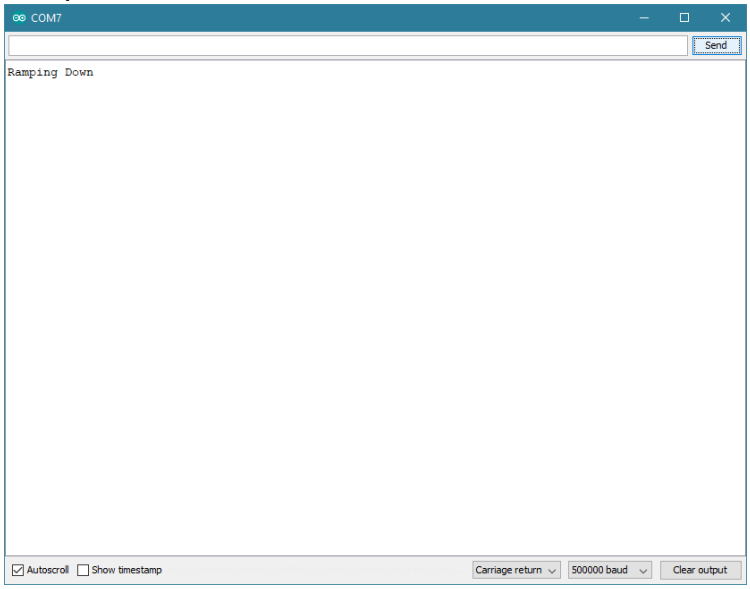
Click on the Upload arrow. After the program compiles and uploads, under the Tools menu, start the Serial Monitor.



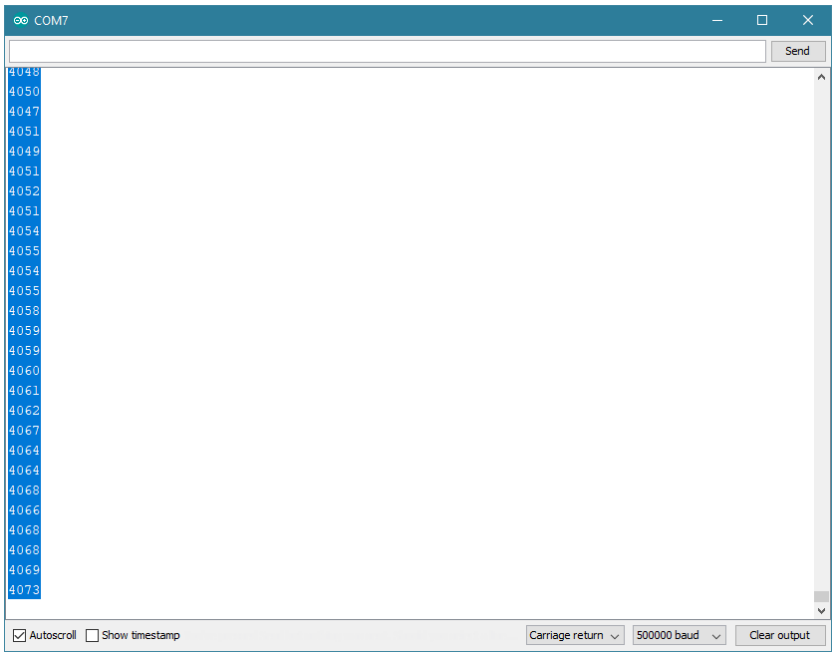
Send a lower case ‘r’ to start the test.



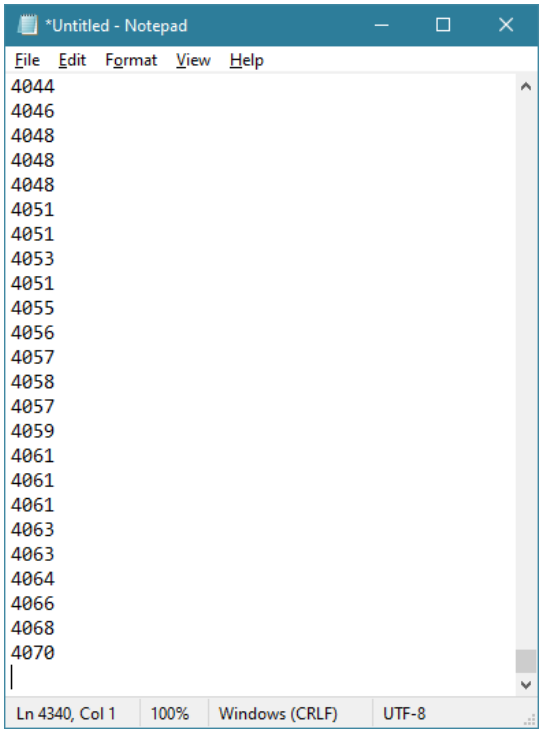
The Pico will indicate the start of the down ramp. It takes a while before the up ramp will start.



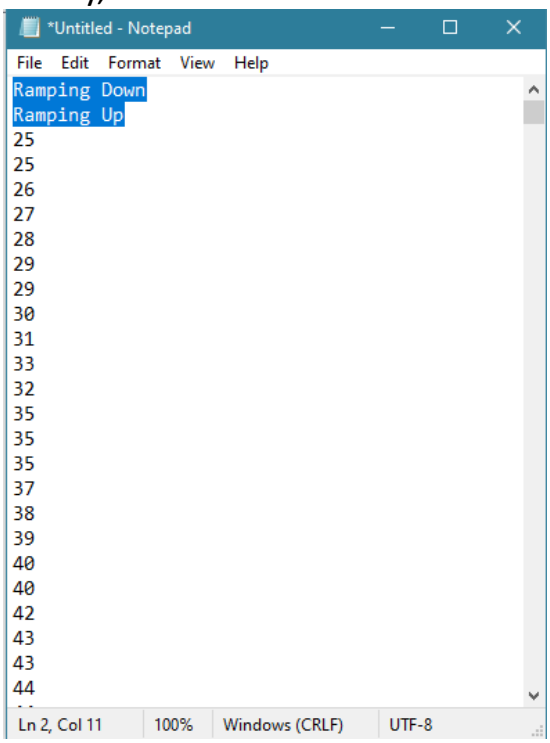
The up ramp will send samples every 5ms until the data exceeds 4070. Right click in the text area. Then, press control-A to select all the text. Next, press control-C to copy it.



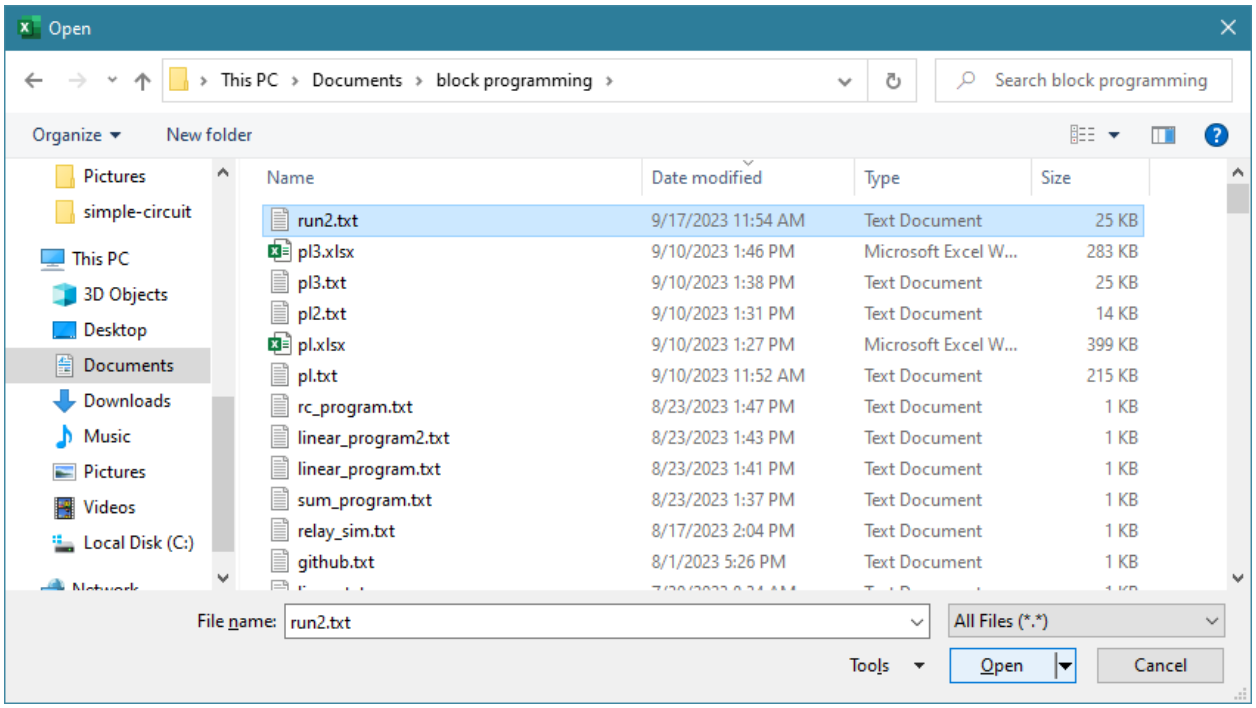
Start Notepad. Select Edit then Paste. Notepad will now contain the test data text.



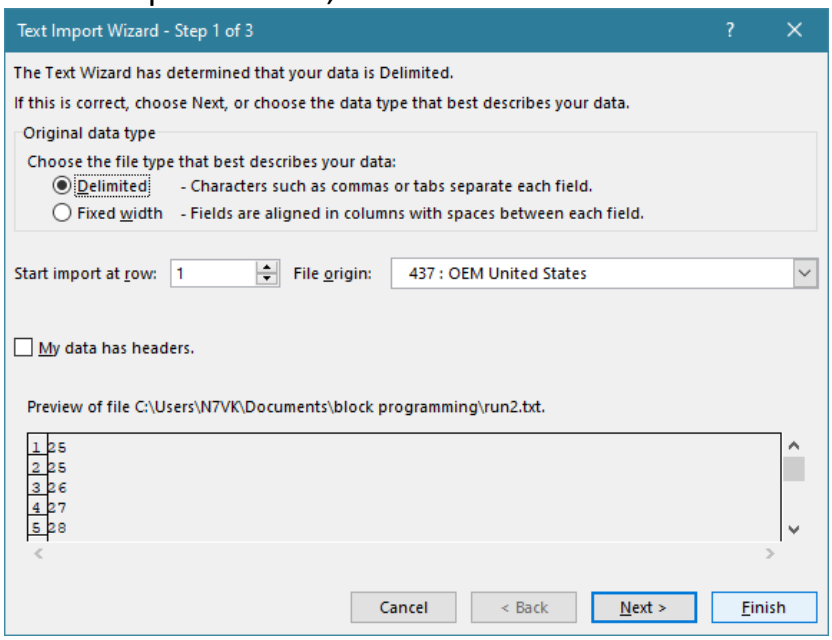
Scroll to the top. Select the ramping text and delete it.
Finally, save the text to a file *runN.txt* where N is your test run number.



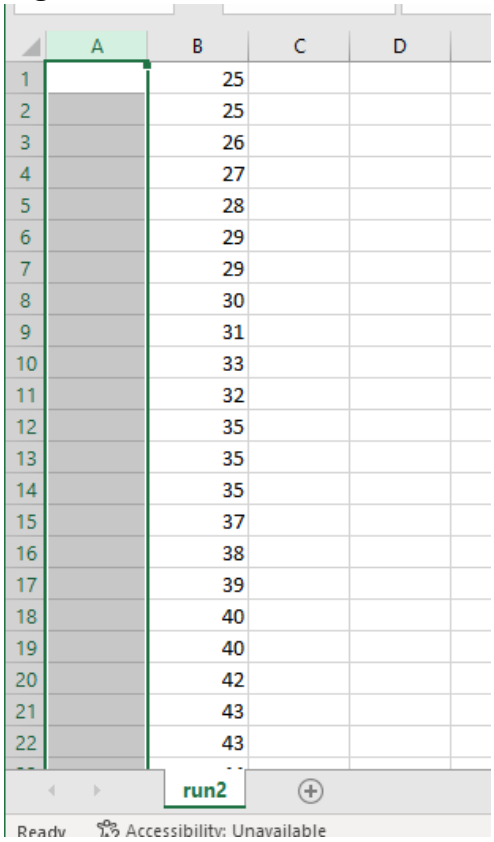
- Start Excel.
- Click on Open then Browse.
- Goto your folder where the text file is saved. Select file type as all.
- Click on your file name then click on Open.



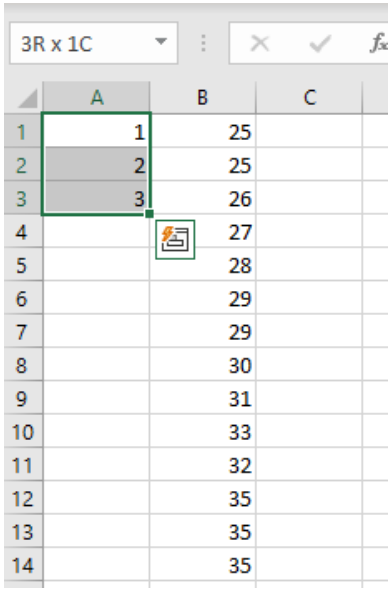
In the Import Wizard, click on finish.



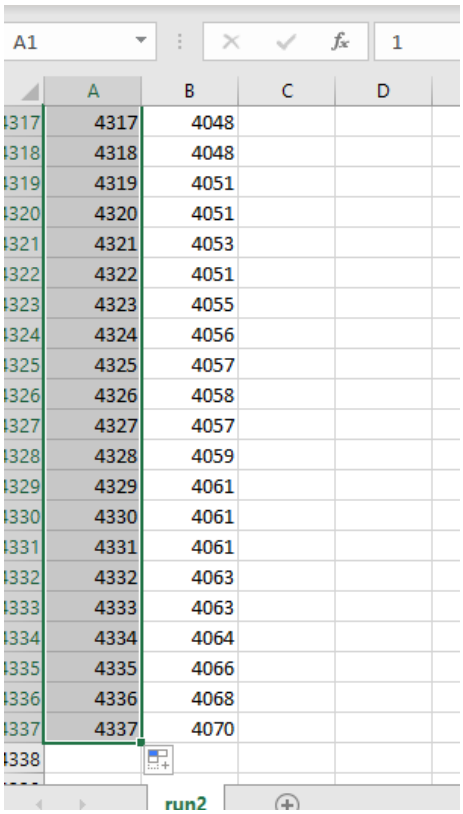
Right Click on the A column and select Insert.



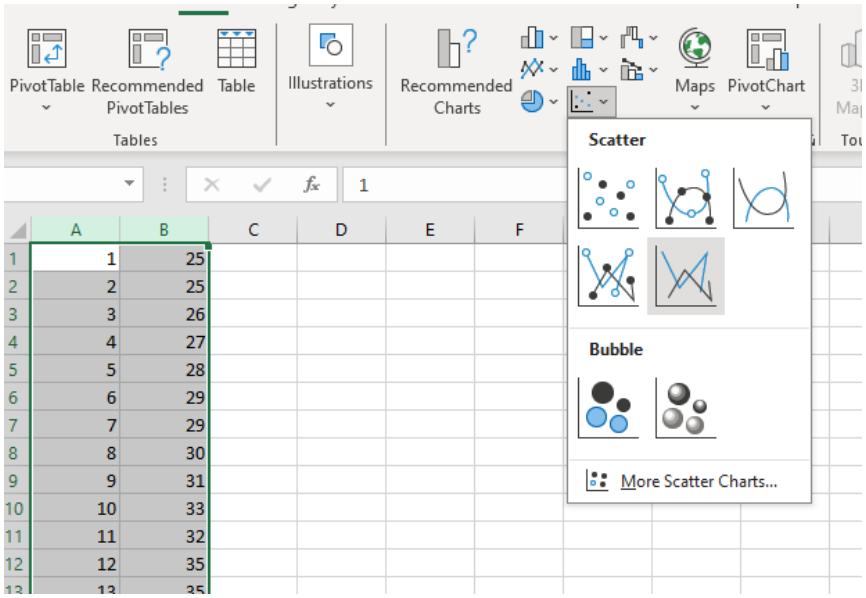
In the A column start a series by entering 1, 2 and 3.



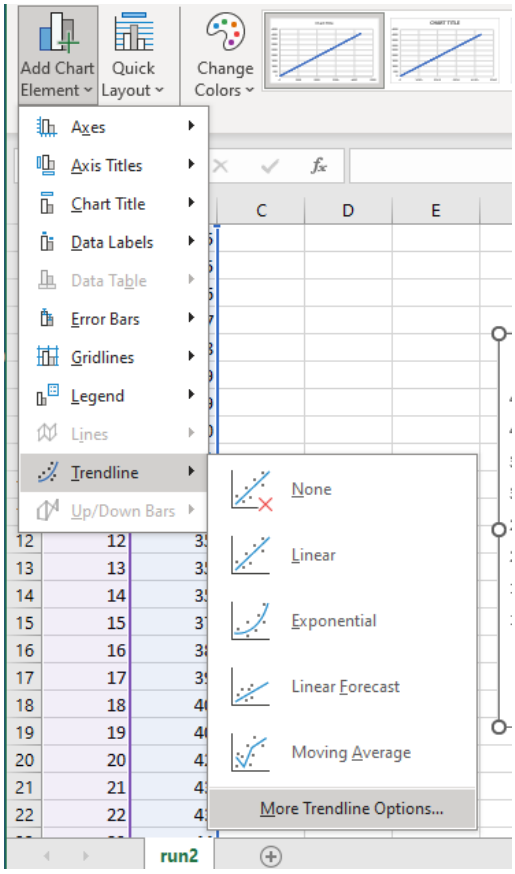
Select the three values. Left click on the lower right corner and drag down to fill the A column with the series. Stop at the last ADC value.



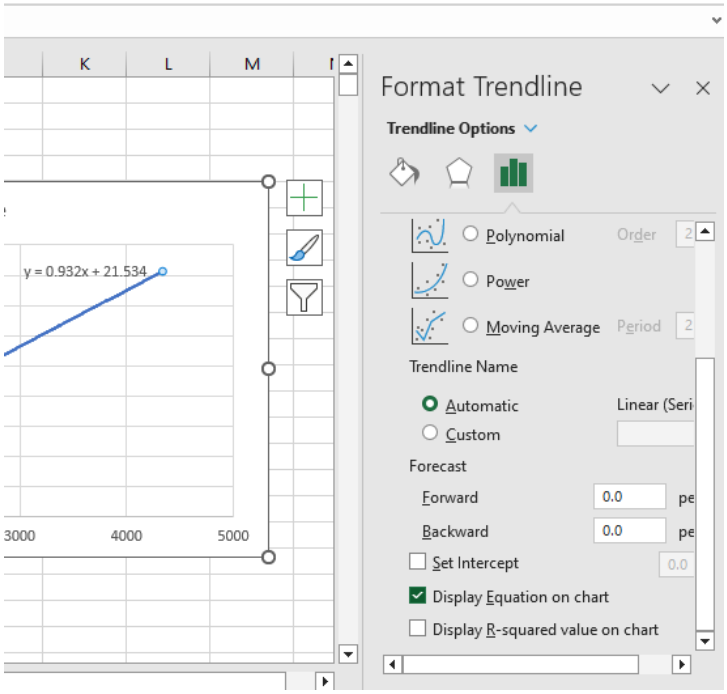
Hold the control key and select the A and B columns. Select Insert and use the straight-line scatter chart.



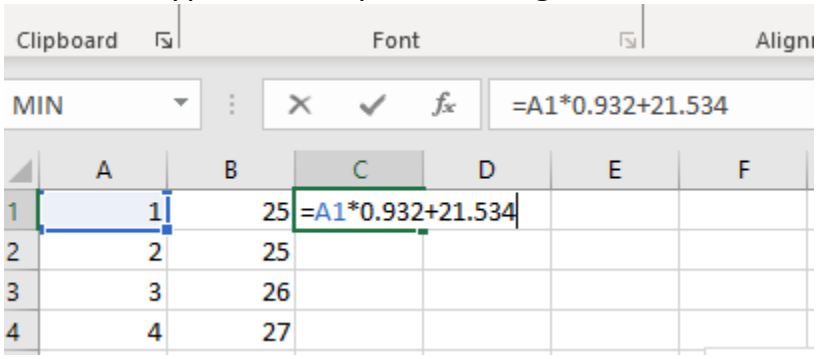
Now add a Chart Element and choose Trendline with more options.



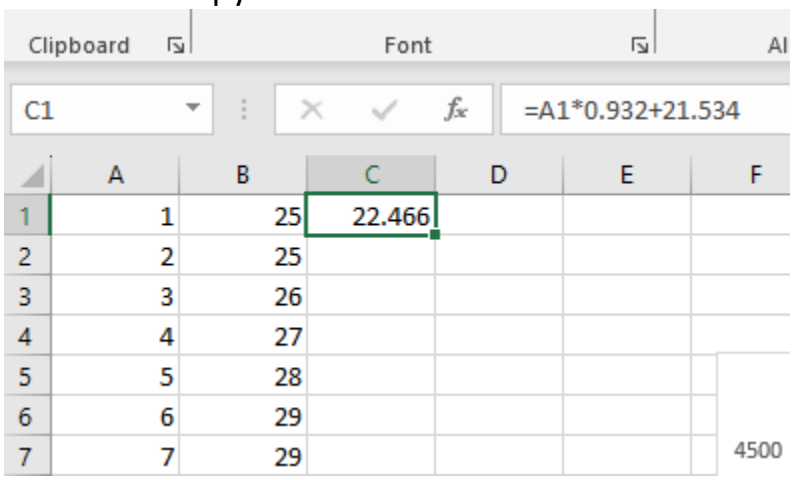
You want to display a linear trend with the equation shown on the chart.



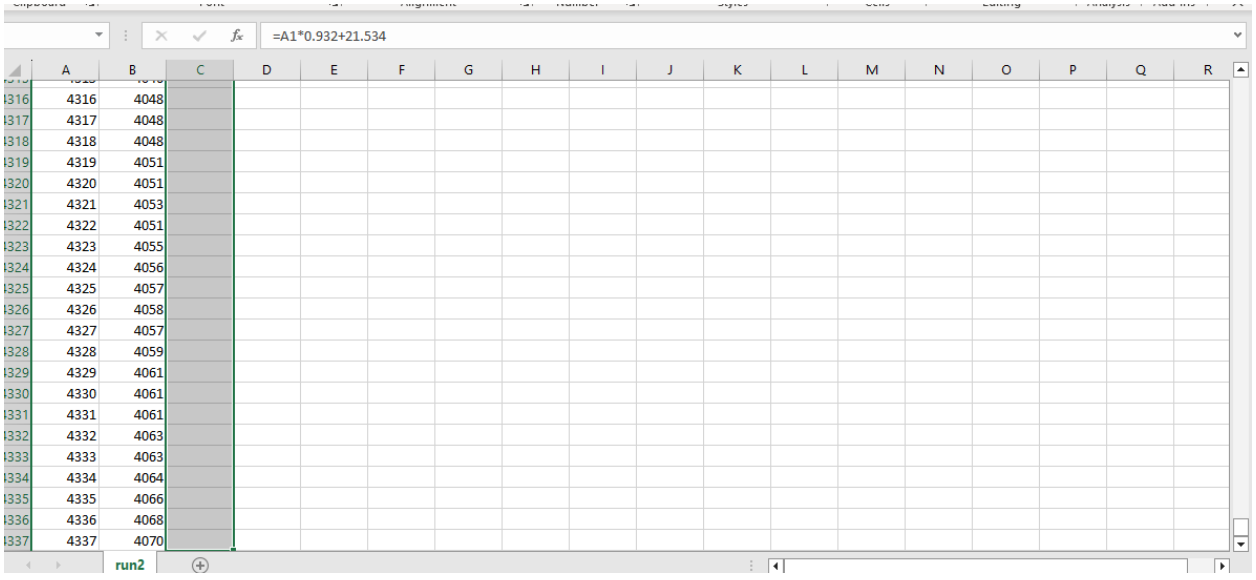
In cell C1 type in the equation using A1 substituted for X.



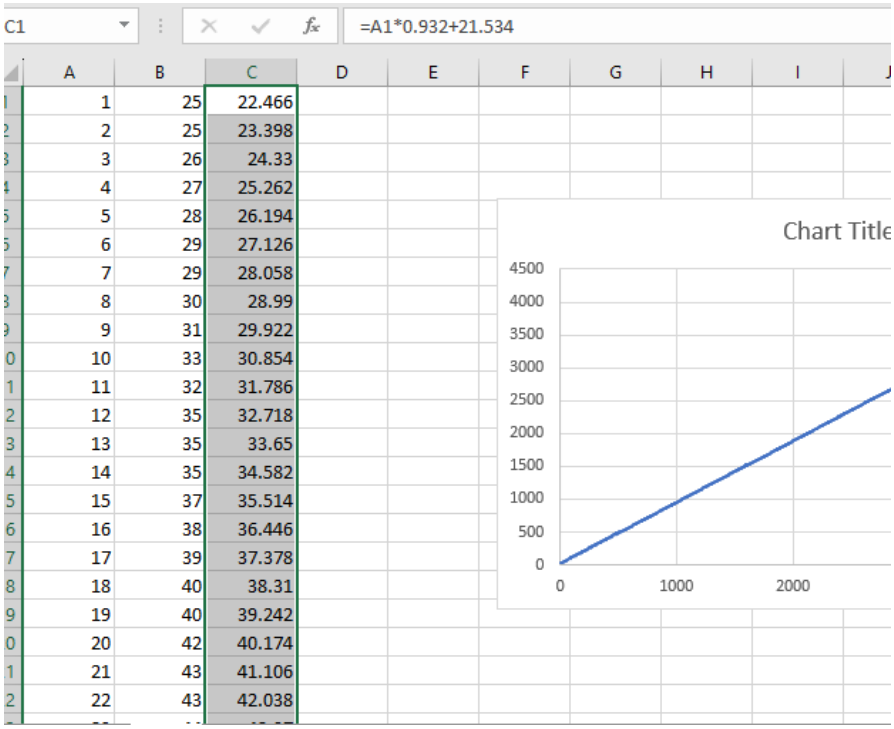
Select and copy the C1 cell.



Use the scroll bar to find the bottom of the data. Press shift and right click in the bottom C column.

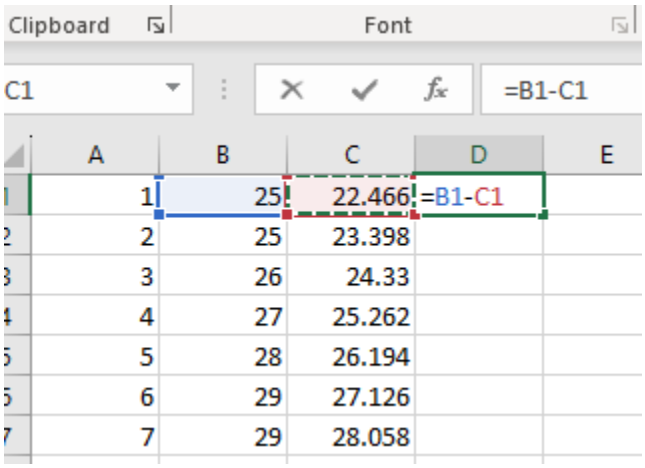


Press control-V to paste the equation into the C column.



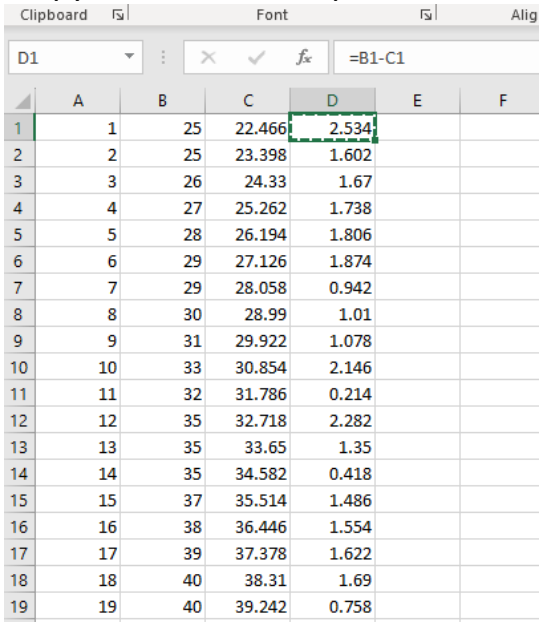
	A	B	C	D	E	F	G	H	I
1	1	25	22.466						
2	2	25	23.398						
3	3	26	24.33						
4	4	27	25.262						
5	5	28	26.194						
6	6	29	27.126						
7	7	29	28.058						
8	8	30	28.99						
9	9	31	29.922						
10	10	33	30.854						
11	11	32	31.786						
12	12	35	32.718						
13	13	35	33.65						
14	14	35	34.582						
15	15	37	35.514						
16	16	38	36.446						
17	17	39	37.378						
18	18	40	38.31						
19	19	40	39.242						
20	20	42	40.174						
21	21	43	41.106						
22	22	43	42.038						

In cell D1 enter an equation to calculate the difference between B1 and C1. This is the error between the ideal line and the measured data.



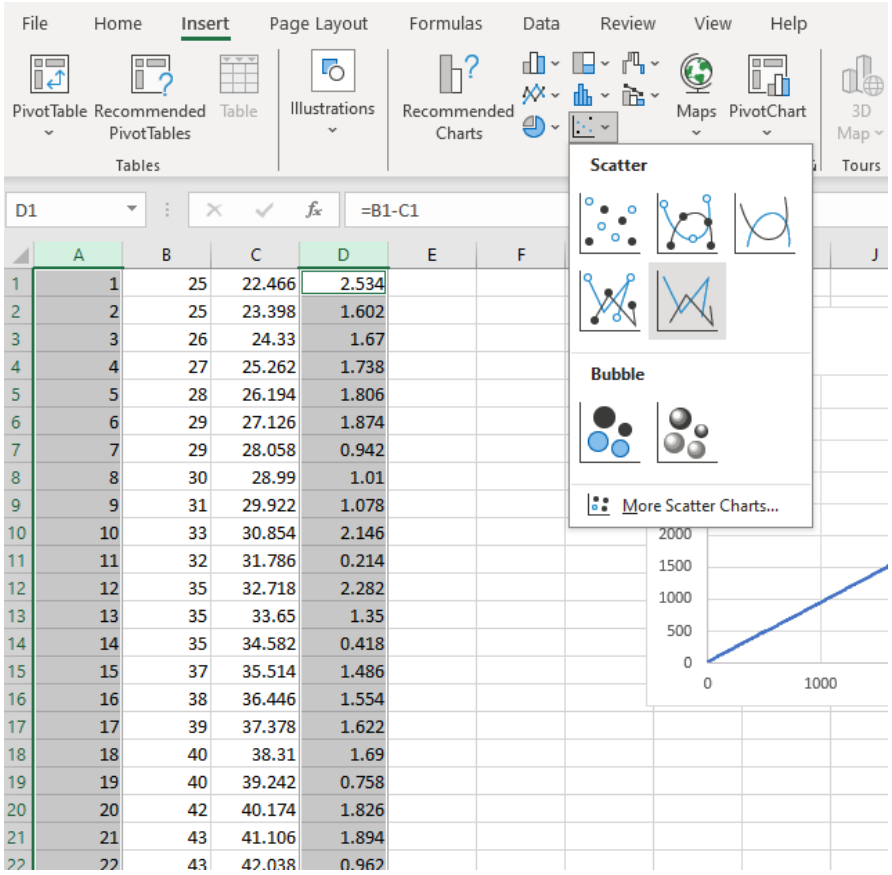
	A	B	C	D	E
1	1	25	22.466	=B1-C1	
2	2	25	23.398		
3	3	26	24.33		
4	4	27	25.262		
5	5	28	26.194		
6	6	29	27.126		
7	7	29	28.058		

Copy the D1 cell and paste it into the D1 column.



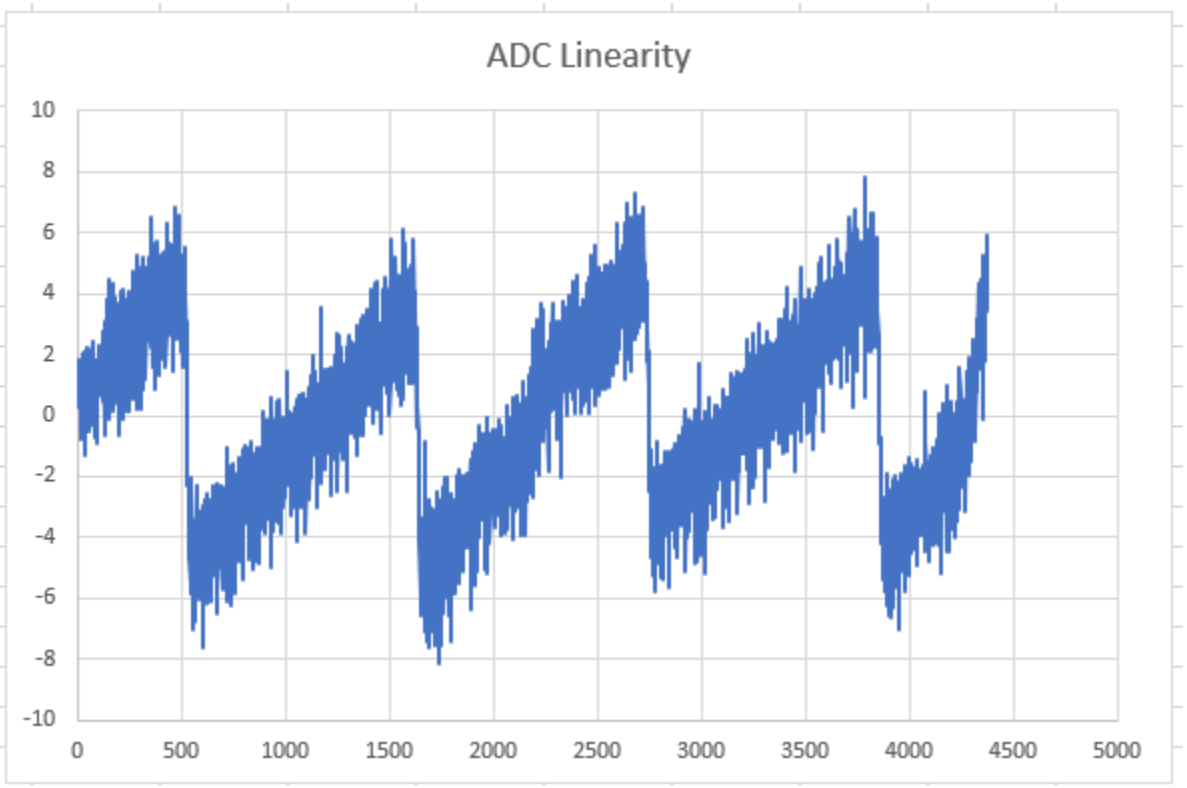
	A	B	C	D	E	F
1	1	25	22.466	2.534		
2	2	25	23.398	1.602		
3	3	26	24.33	1.67		
4	4	27	25.262	1.738		
5	5	28	26.194	1.806		
6	6	29	27.126	1.874		
7	7	29	28.058	0.942		
8	8	30	28.99	1.01		
9	9	31	29.922	1.078		
10	10	33	30.854	2.146		
11	11	32	31.786	0.214		
12	12	35	32.718	2.282		
13	13	35	33.65	1.35		
14	14	35	34.582	0.418		
15	15	37	35.514	1.486		
16	16	38	36.446	1.554		
17	17	39	37.378	1.622		
18	18	40	38.31	1.69		
19	19	40	39.242	0.758		

Press and hold the control key. Select column A and column D. Then insert a straight-line scatter chart.



	A	B	C	D	E	F
1	1	25	22.466	2.534		
2	2	25	23.398	1.602		
3	3	26	24.33	1.67		
4	4	27	25.262	1.738		
5	5	28	26.194	1.806		
6	6	29	27.126	1.874		
7	7	29	28.058	0.942		
8	8	30	28.99	1.01		
9	9	31	29.922	1.078		
10	10	33	30.854	2.146		
11	11	32	31.786	0.214		
12	12	35	32.718	2.282		
13	13	35	33.65	1.35		
14	14	35	34.582	0.418		
15	15	37	35.514	1.486		
16	16	38	36.446	1.554		
17	17	39	37.378	1.622		
18	18	40	38.31	1.69		
19	19	40	39.242	0.758		
20	20	42	40.174	1.826		
21	21	43	41.106	1.894		
22	22	43	42.038	0.962		

Modify the chart to your preferences. From the data, it is apparent the ADC has about 8 least significant bits of non-linearity. The Pico 12-bit converter is specified at 9-bits and it meets this requirement.



Some improvement in linearity is possible. You can read about the technique in the reference.

https://github.com/simple-circuit/Block-Diagram-Compiler

file: ADC Linearity.pdf