Generating Sine Waves with PWGs

Presenters: Noah Breedy, Camille Catolos, Sarah Liu

1/13/2023

@BrookhavenLab

About Us

Camille Catolos, Skyline College, California
Major: Computer Science

Noah Breedy, SUNY Polytechnic University, New York
Major: Computer Science

Sarah Liu, The City College of New York.
Major: Mechanical engineering

Problem

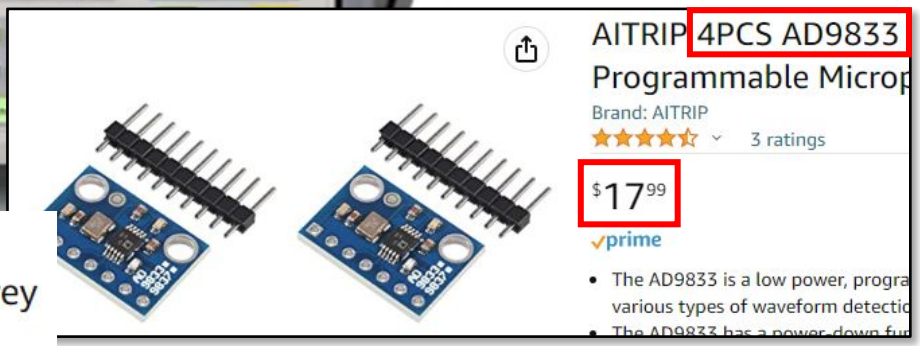
Modeling an sine wave maker using the AD9833 signal generator as an cheaper alternative to other products in the market.

Siglent Technologies SDG2042X Arbitrary Waveform Function-Generators, 40 MHz, Grey

Visit the Siglent Technologies Store

97 ratings | 30 answered questions

$499.00



AITRIP 4PCS AD9833 Programmable Microp

Brand: AITRIP

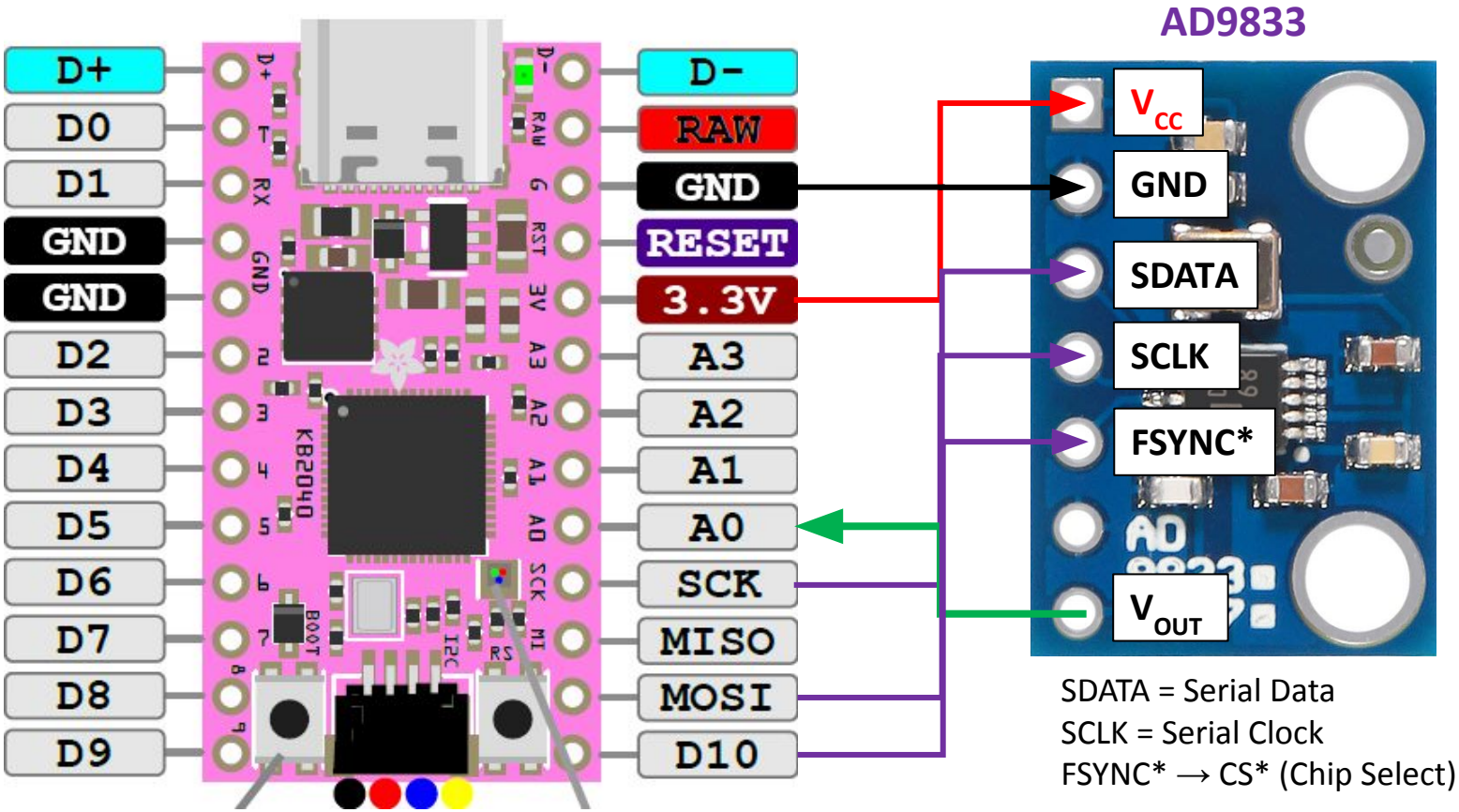
3 ratings

$17.99

prime

- The AD9833 is a low power, progra various types of waveform detectio
- The AD9833 has a power-down fu

The sig_gen Circuit



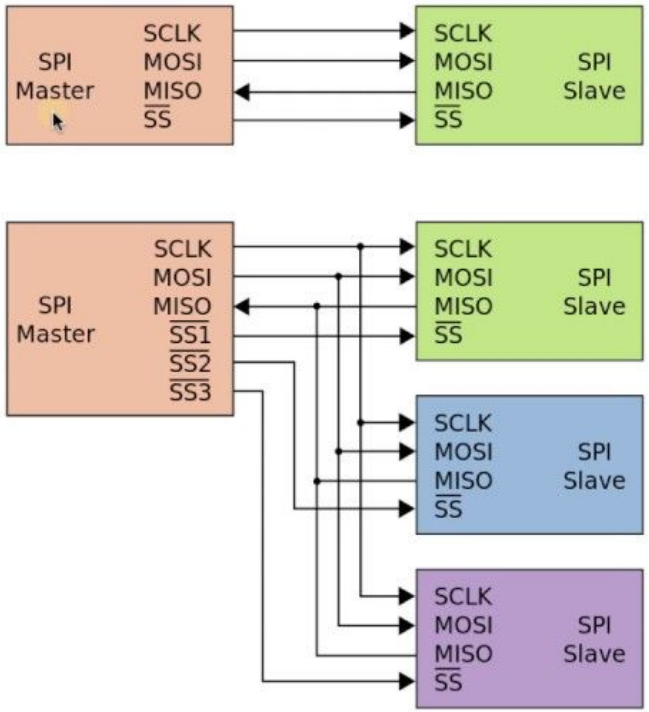
SDATA = Serial Data
SCLK = Serial Clock
FSYNC* ⟶ CS* (Chip Select)

MISO ⟶ POCI (Peripheral Out / Controller In)
MOSI ⟶ PICO (Peripheral In / Controller Out

GPIO Pin A0 will be set in **ADC** mode

Serial Peripheral Interface (SPI)

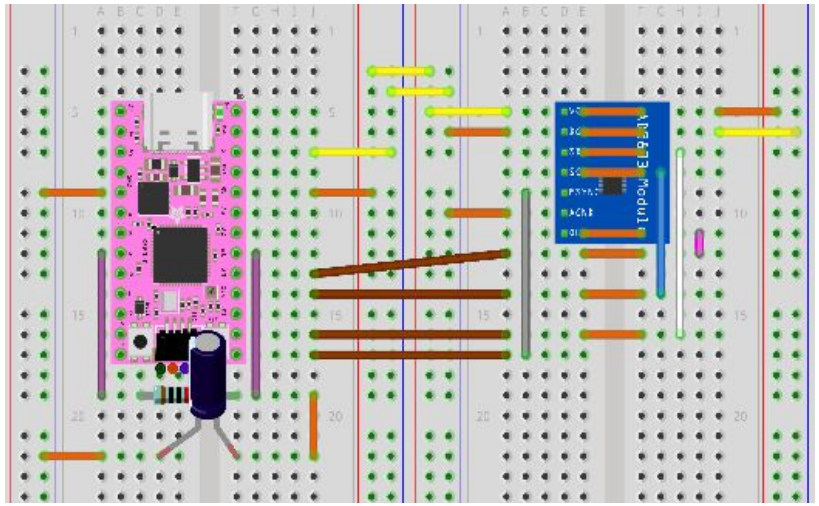
- **Synchronous** serial communication interface specification used for **short distance communication** in embedded systems.
- Developed by **Motorola** in the **mid 1980s** and has become a **de facto standard** (achieved a dominant position by public acceptance).
- Full duplex mode using a (**single master–multi slaves**) architecture.
- Used to interface sensors, memories displays, and Secure Digital (SD) cards.



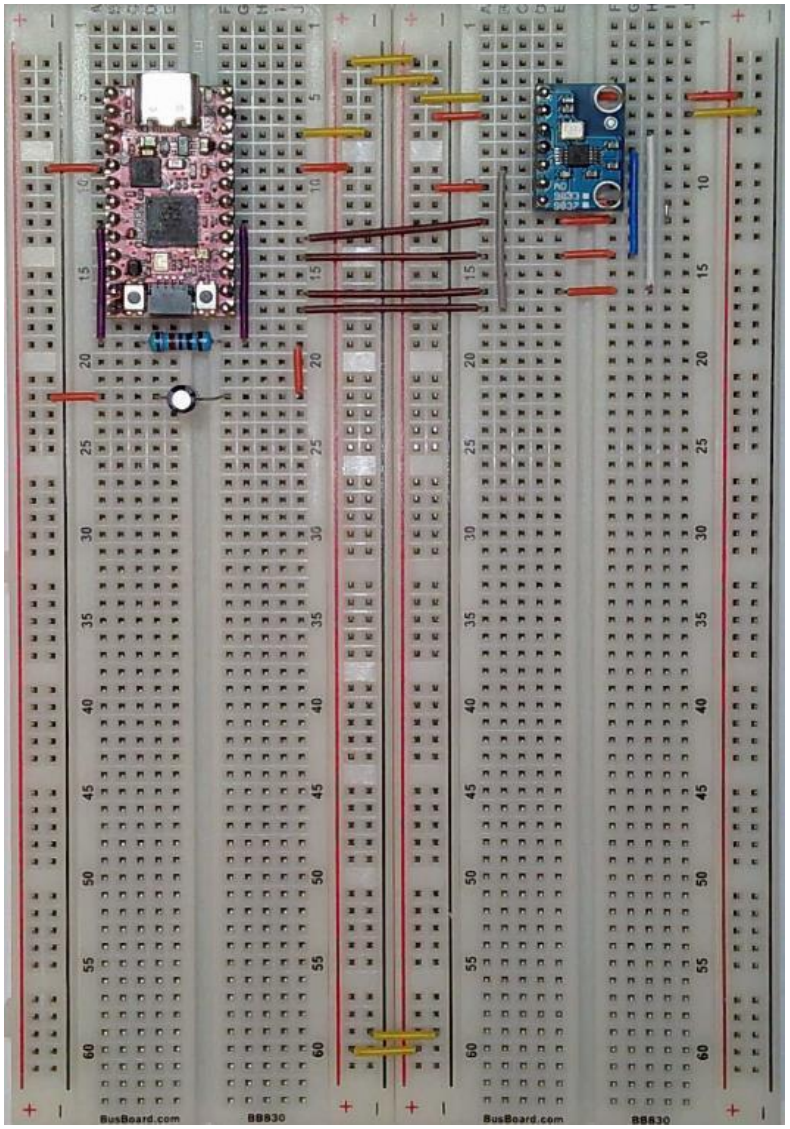
MISO → POCI (Peripheral Out / Controller In)
MOSI → PICO (Peripheral In / Controller Out

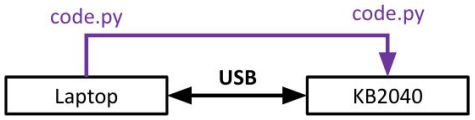
Strategy

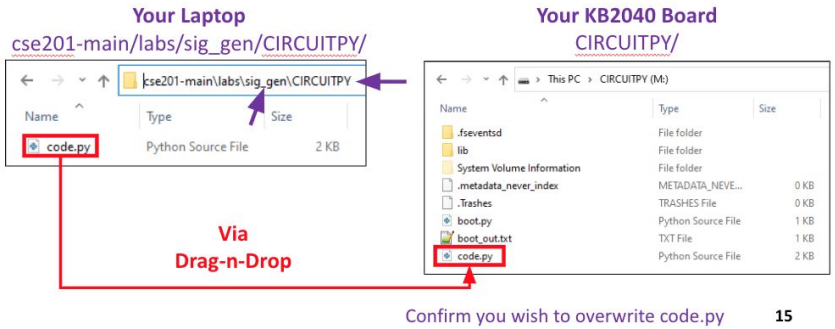
Using the materials given, we created a circuit that would model the communication between our microcontroller and signal generator.



Fritzing Image







sig_gen.py : file that commands the circuit board

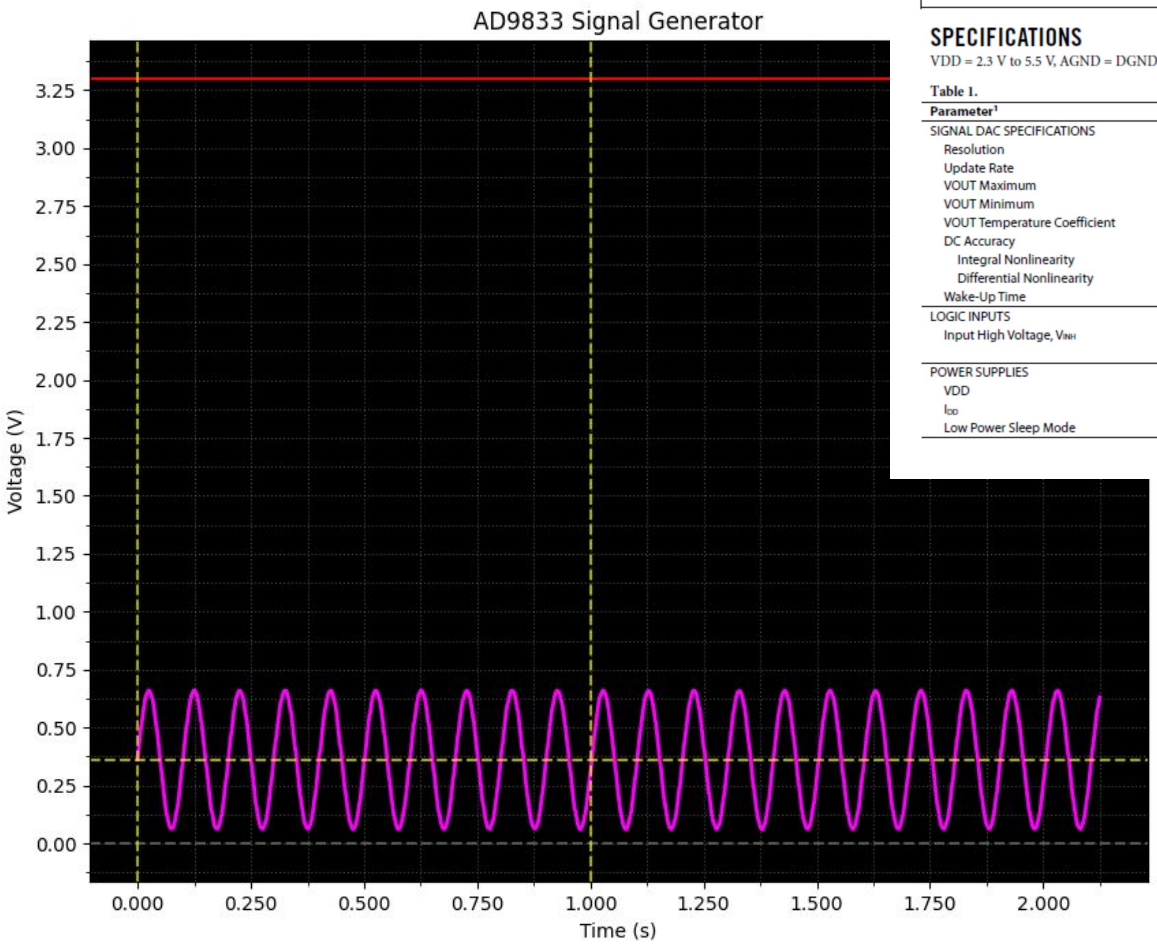
```
# recieves data from code.py
def usb_readline(usb_data_port):
    s = usb_data_port.readline().decode("ASCII").strip()
    return s


def main():

    ser = serial.Serial(None, 115200, 8, "N", 1, timeout=120) # opens up the data port

    try:
        port = "COM5"
        if sys.platform == "linux":
            port = "/dev/ttyAMA0"
        if sys.platform == "darwin":
            port = "/dev/tty.usbserial-110"

        ser.port = port
        ser.open()

        # Send MCU the command to (r)un the experiment
        ser.write(b"r\n")
        print("Signal Generator experiment is running...")



        # Read number of samples from USB
        n = int(usb_readline(ser))

        # Declare numpy arrays to store the samples
        times = np.zeros(n, float)
        volts = np.zeros(n, float)

        # Read times and volts samples from USB into arrays
        for i in range(n):
            times[i] = int(usb_readline(ser))
        for i in range(n):
            volts[i] = int(usb_readline(ser))
```

```
def read_samples():
    pixel_builtin.fill((255, 0, 0))  # RED  ← ①

    # Set number of samples (NOT number of seconds!)
    n = 2000  ← ②
    volts = [int] * n
    times = [int] * n  ← ③

    # Start generating 10 Hz sine waves
    wave_gen.reset()  ← ⑤
    wave_gen.update_freq(10)  # Hz  ← ④
    wave_gen.start()  ← ⑥

    # Read voltage samples
    for i in range(n):
        times[i] = time.monotonic_ns()
        volts[i] = pin_adc.value  ← ⑦
        time.sleep(0.001)  ← ⑧

    # Transfer data over USB data port
    pixel_builtin.fill((255, 255, 0))  # YELLOW
    usb_writeline(ser, n)  # number of samples
    for val in times:  ← ⑨
        usb_writeline(ser, val)  # times array
    for val in volts:  ← ⑩
        usb_writeline(ser, val)  # volts array
    pixel_builtin.fill((0, 255, 0))  # GREEN
```

```
while True:
    cmd = ser.readline().strip().decode("utf-8")
    if cmd == "r":
        read_samples()
```

code.py : file that connects the laptop & the KB2040

Result



Analog Devices AD9833 **Datasheet**

Data Sheet AD9833

SPECIFICATIONS

VDD = 2.3 V to 5.5 V, AGND = DGND = 0 V, $T_A = T_{MIN}$ to T_{MAX}, R_{SET} = 6.8 kΩ for VOUT, unless otherwise noted.

Table 1.

Parameter[1]	Min	Typ	Max	Unit	Test Conditions/Comments
SIGNAL DAC SPECIFICATIONS					
Resolution		10		Bits	
Update Rate			25	MSPS	
VOUT Maximum		0.65		V	
VOUT Minimum		38		mV	
VOUT Temperature Coefficient		200		ppm/°C	
DC Accuracy					
Integral Nonlinearity		±1.0		LSB	
Differential Nonlinearity		±0.5		LSB	
Wake-Up Time		1		ms	
LOGIC INPUTS					
Input High Voltage, V_{INH}	1.7			V	2.3 V to 2.7 V power supply
	2.0			V	2.7 V to 3.6 V power supply
POWER SUPPLIES					f_{MCLK} = 25 MHz, $f_{OUT} = f_{MCLK}/4096$
VDD	2.3		5.5	V	
I_{DD}		4.5	5.5	mA	I_{DD} code dependent; see Figure 7
Low Power Sleep Mode		0.5		mA	DAC powered down, MCLK running

Rev. G | Page 3 of 21

Thank You

We appreciate all the support and help throughout this program

Section Break