ADC Interface with Raspberry Pi 4

Anirudh Ashrit, 014535846
Electrical Engineering Department, College of Engineering
San Jose State University, San Jose, CA 94303
E-mail: anirudh.ashrit@sjsu.edu

Abstract – Analog to Digital Converter (ADC) is a device which converts the analog signals received as input to corresponding digital signal as output. It is a quintessential device in electronic circuits and used widely in IoT applications. In this experiment, an ADC is interfaced with a Raspberry board to overcome its absence and observe its functionalities.

Keywords – ADC, Pi 4, Potentiometer

1. Introduction

ADCs convert an analog input primarily voltage to a digital signal. Different values like temperature, humidity and speed are all converted into voltage using sensors and ADC in-turn converts this voltage into a digital signal comprised of 1's and 0's. The operation of ADC heavily relies on the channel and resolution (bits). ADS1015 ADC used in the current experiment is a 4-channel and 12-bit device, meaning, there are 4 pins on the device which can read voltage with a resolution of 12 bits each. The resolutions are adjusted and mapped based on the voltage range of the ADC. For example, if the voltage range is 0-10V then, ADC reads 0V and produces 0 as output and for 10V, the output is 4095 (2^{12} = 4096). Similarly, for an input between the range, i.e. 5V, the output is assigned nearly as 2047. From above statements one can conclude that output is expected to be linear as shown in Fig. 1

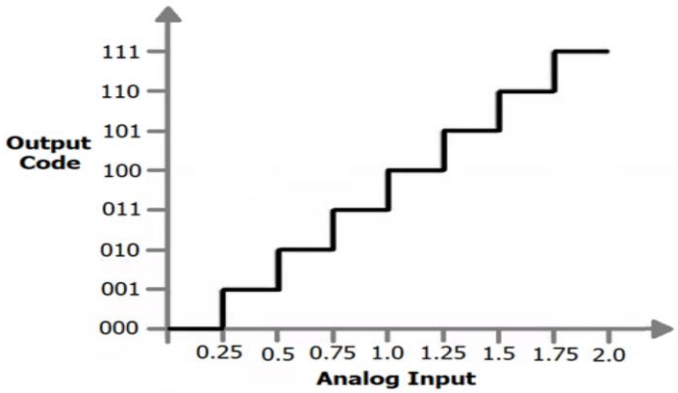

Fig.1 Characteristics of a 3-bit ADC with 2V range

The same is achieved in this project using a potentiometer for varied voltages and their corresponding responses. Below mentioned formula is implemented for determining a digital value to the corresponding analog value.

(ADC Resolution / Operating Voltage) = (ADC Digital Value / Actual Voltage Value)

2. Hardware and Software

The hardware implemented in achieving results are a Raspberry Pi 4B board, Potentiometer, Jumper Wires, ADC, RJ45 cable. Device's operation is executed in Raspbian OS using Python language.

2.1. Connecting the circuit

The pins of Pi 4B, Potentiometer and ADC are connected with respect to their datasheet. Code written in Python is executed using Raspbian OS and results are obtained.

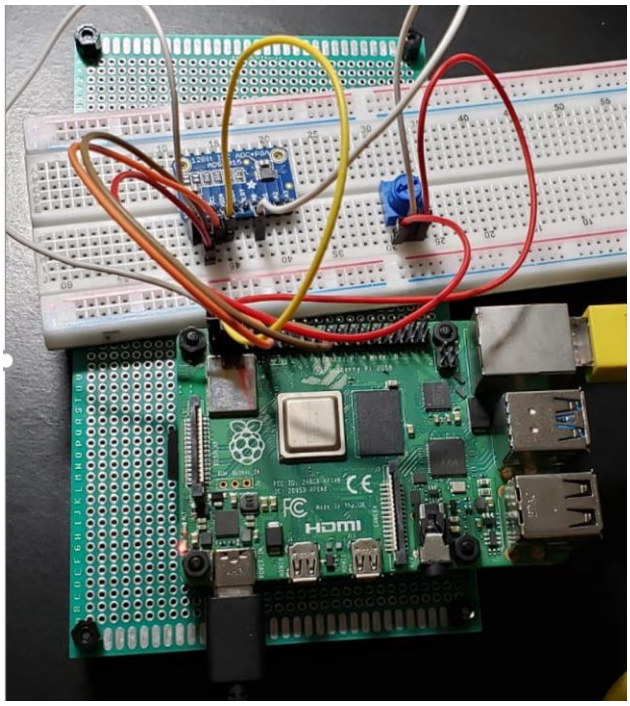

Fig. 2 Circuit of ADC interface with Pi 4B

3. Testing and Verification

After performing the above discussed connections, the ADC was successfully tested and was observed to have produced the following results:

0	1	2	3
0	0	0	0
0	0	0	0
0	0	0	0
28	0	0	0
86	0	0	0
76	0	0	0
91	0	0	0
173	0	0	0
252	0	0	0
321	0	0	0
434	0	0	0
454	0	0	0
574	0	0	0

Fig. 3 Output of ADC

574	0	0	0
645	0	0	0
809	0	0	0
868	0	0	0
877	0	0	0
896	0	0	0
1034	0	0	0
1155	0	0	0
1197	0	0	0
1233	0	0	0
1249	0	0	0
1252	0	0	0
1446	0	0	0
1547	0	0	0
1587	0	0	0
1724	0	0	0
1837	0	0	0
1877	0	0	0
1877	0	0	0
2047	0	0	0
2047	0	0	0
2047	0	0	0
2047	0	0	0
2047	0	0	0
2047	0	0	0

Fig. 4 Output of ADC for channel_0

4. Conclusion

Voltage was given as input and ADC successfully converted the result into a digital value by mapping the values of provided range. Hence, the experiment was successfully completed.

5. References

[1] H. Li, "Author Guidelines for CMPE 146/242 Project Report", *Lecture Notes of CMPE 146/242*, Computer Engineering Department, College of Engineering, San Jose State University, March 6, 2006, pp. 1.

[2] Circuit Digest – What is ADC ? https://circuitdigest.com/tutorial/what-is-adc-analog-to-digital-converters

[3] https://www.slideshare.net/sameeeee/ieee-paper-format

Appendix

```
import time

# Import the ADS1x15 module.
import Adafruit_ADS1x15

# Create an ADS1015 ADC (12-bit)
instance.
adc = Adafruit_ADS1x15.ADS1015()

# Note you can change the I2C address from
its default (0x48), and/or the I2C
# bus by passing in these optional
parameters:
#adc
=Adafruit_ADS1x15.ADS1015(address=0x
49, busnum=1)

# Choose a gain of 1 for reading voltages
from 0 to 4.09V.
# Or pick a different gain to change the
range of voltages that are read:
#  - 2/3 = +/-6.144V
#  -   1 = +/-4.096V
#  -   2 = +/-2.048V
#  -   4 = +/-1.024V
#  -   8 = +/-0.512V
#  -  16 = +/-0.256V
# See table 3 in the ADS1015/ADS1115
datasheet for more info on gain.
GAIN = 1

print('Reading ADS1x15 values, press Ctrl-
C to quit...')
# Print nice channel column headers.
print('| {0:>6} | {1:>6} | {2:>6} | {3:>6}
|'.format(*range(4)))
print('-' * 37)
# Main loop.
while True:
    # Read all the ADC channel values in a
list.
    values = [0]*4
    for i in range(4):
        # Read the specified ADC channel
using the previously set gain value.
        values[i] = adc.read_adc(i,
gain=GAIN)

    # Print the ADC values.
    print('| {0:>6} | {1:>6} | {2:>6} | {3:>6}

|'.format(*values))

# Pause for half a second.
    time.sleep(0.5)
```