**Lab 3: Analog Input Operations**

**Objectives**

* Introduce students to Analog Input Operations in Raspberry Pi
* Interfacing sensors to Raspberry Pi
* Introduction to ADC Pi plus

**Equipment and Tools**

* Monitor
* HDMI to VGA converter
* IDLE IDE
* Raspberry Pi 3 with Raspbian Installed
* Micro USB cable
* ADC Pi Plus
* HIH -4000 Relative humidity sensor : <http://www.phanderson.com/hih-4000.pdf>
* TLC272 Op-Amp
* LM35 Temperature Sensor : <http://www.ti.com/lit/ds/symlink/lm35.pdf>
* Wires and resistors

**ADC Pi Plus Introduction**

The ADC Pi Plus [shown in Figure.1] is a module that allows Raspberry Pi to read analog values. ADC Pi Plus is an 8 channel 17 bit analogue to digital converter designed to work with the Raspberry Pi A+, B+, 2 and 3. The ADC Pi Plus is based on two Microchip MCP3424 A/D converters each of which contributes 4 analogue inputs. The two MCP3424 A/D converters use I2C to communicate with the Raspberry Pi, giving you eight analogue inputs to use. The I2C address bits are selectable using the on-board jumpers.  The MCP3424 supports up to 8 different I2C addresses so with two A/D converters on each ADC Pi. Each MCP3424 board takes one I2C address. Thus, one board takes two I2C addresses.

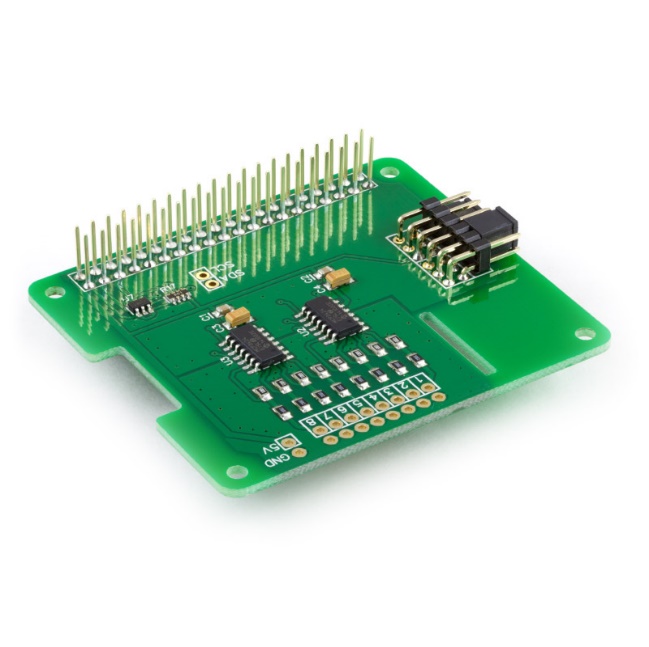


Figure 1. ADC Pi Plus

**Features [1]**

* 8 x 17-bit 0 to 5V Single Ended Inputs
* Uses Raspberry Pi I2C port to communicate
* Stack up to 4 ADC Pi V2 boards on a single Raspberry Pi
* Jumper selectable I2C addresses
* Based on the MCP3424 from Microchip Technologies Inc
* On-Board Programmable Gain Amplifier (PGA): Gains of 1, 2, 4 or 8
* Programmable Data Rate Options:  
     - 3.75 SPS (17 bits)  
     - 15 SPS (15 bits)  
     - 60 SPS (13 bits)  
     - 240 SPS (11 bits)
* One-Shot or Continuous Conversion Options

Figure 2 shows ADC board.

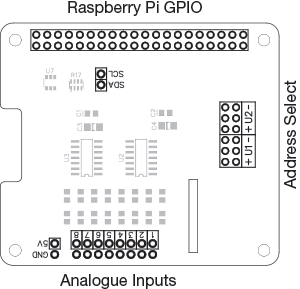


Figure 2. ADC Pi Plus Board Layout

Figure 3 shows Address Select pins of ADC Pi Plus and Table 1 gives guidelines to choose correct I2C Address.

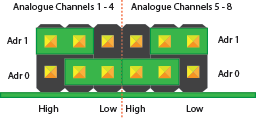
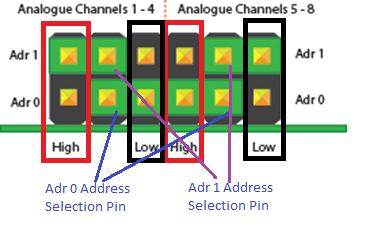
 

Figure 3. Address Selection Pins

For instance, set I2C address of Chip 1 of ADC Pi as 0x6A, we make Adr 0 as Low and Adr 1 as High as shown in Figure. 4.

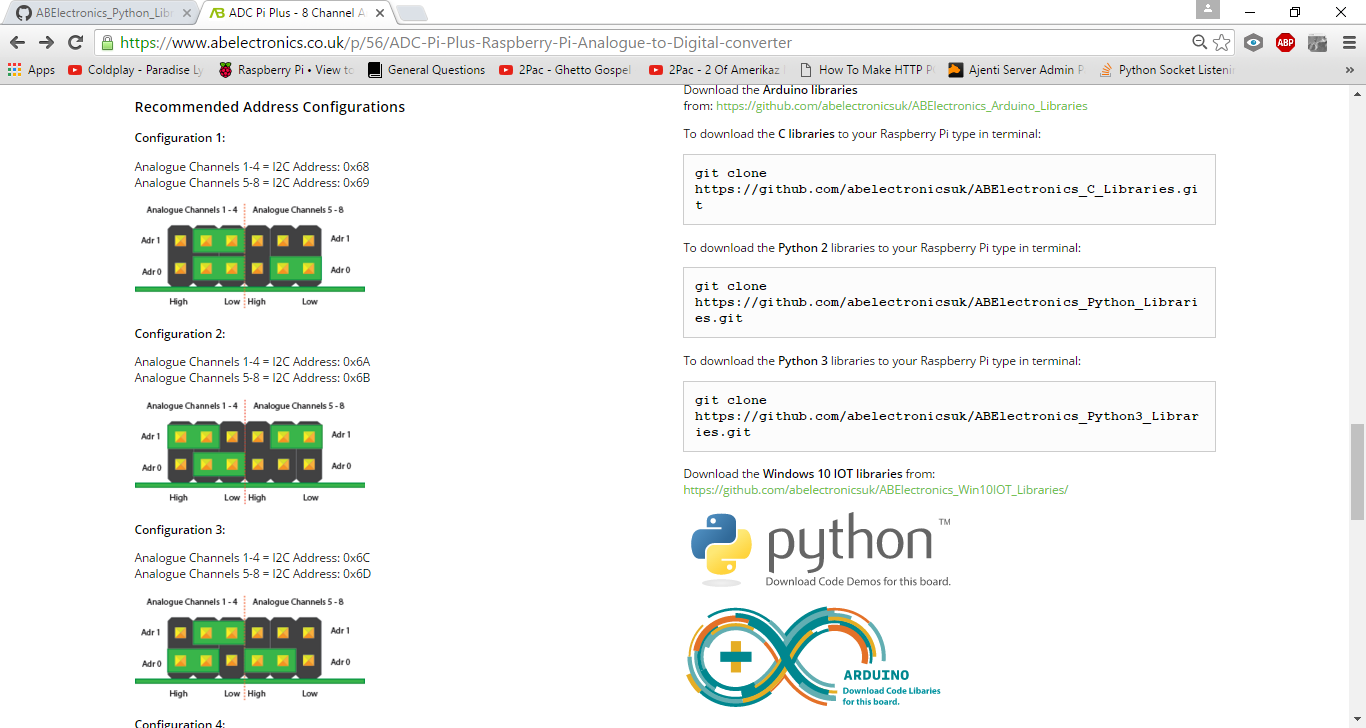
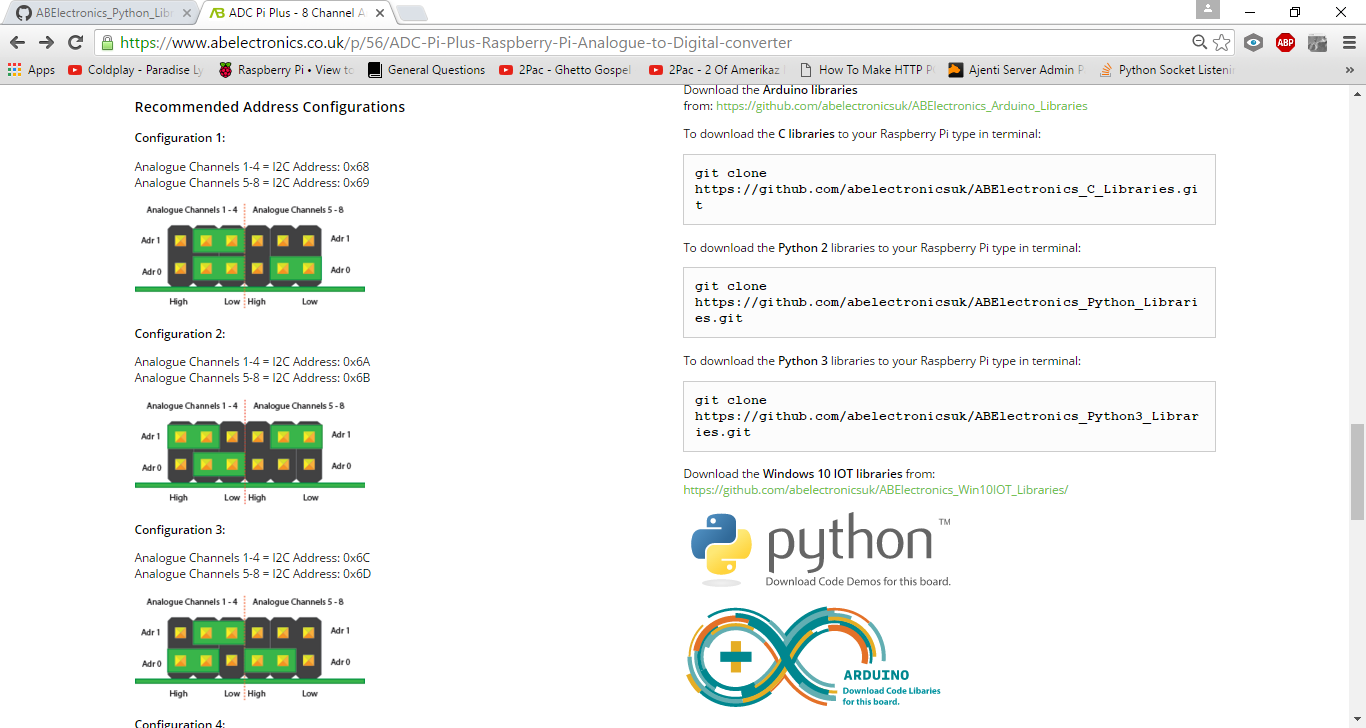


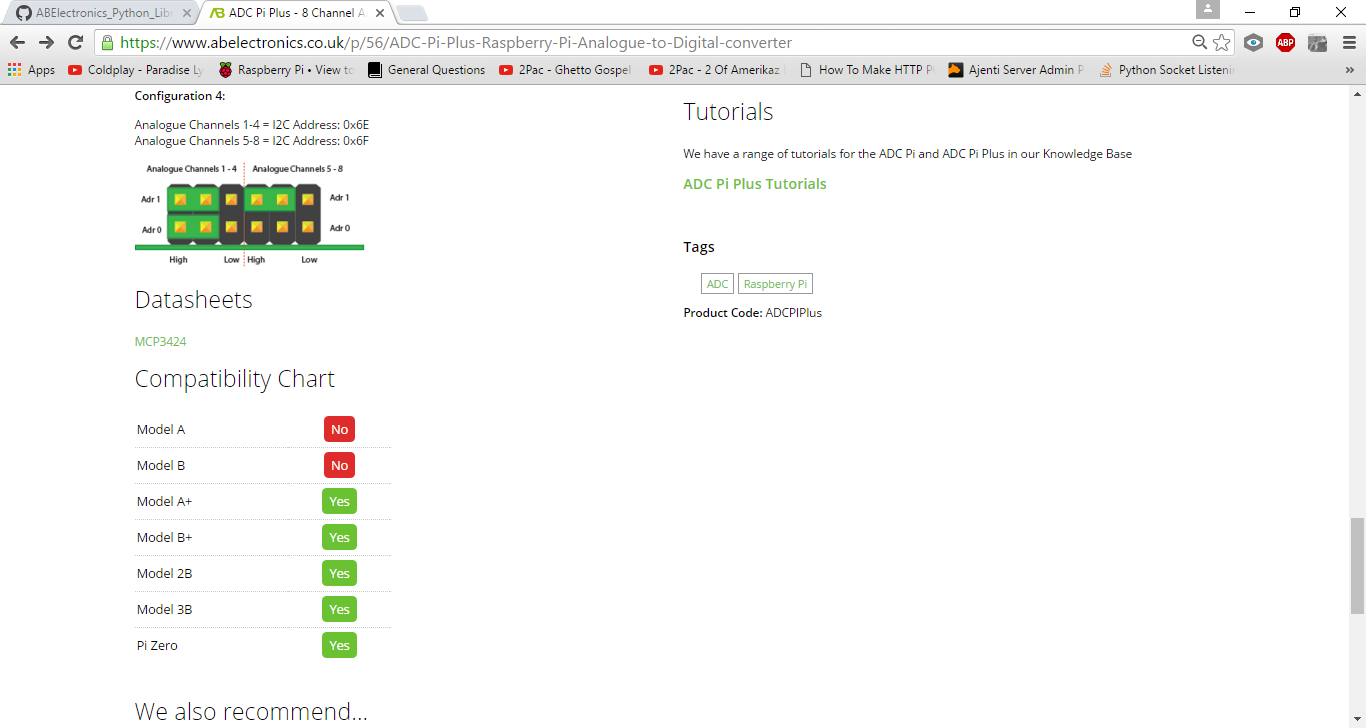
Figure 4. 0x6A configuration 0f I2C Address for Chip 1

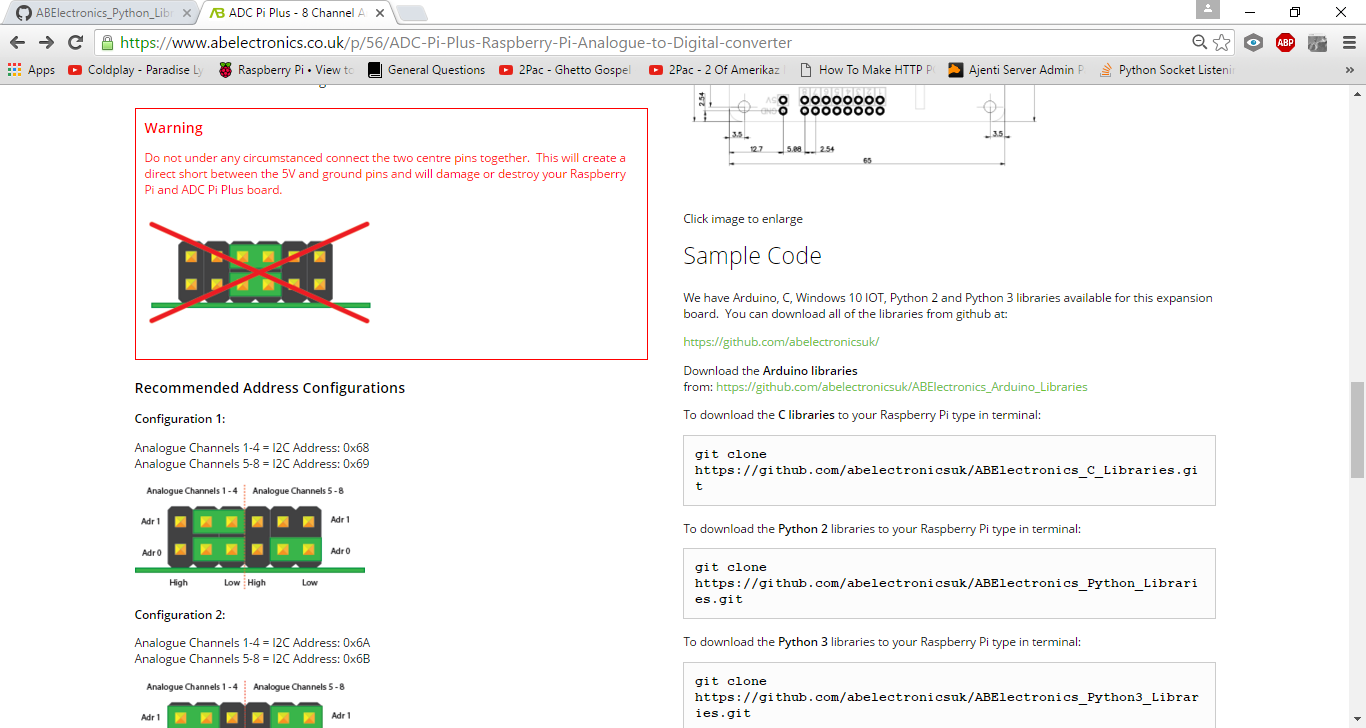
Table 1. I2C Address Configuration

|  |  |  |
| --- | --- | --- |
| Adr 0 | Adr 1 | I2C Address |
| Low | Low | 0x68 |
| Low | Float | 0x69 |
| Low | High | 0x6A |
| Float | Low | 0x6B |
| Float | Float | 0x68 |
| Float | High | 0x6F |
| High | High | 0x6E |
| High | Float | 0x6D |
| High | Low | 0x6C |

**Some Recommended Address Configurations**



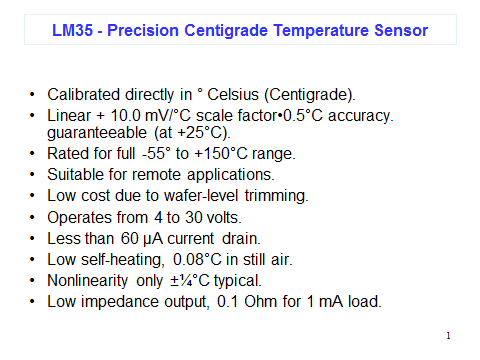




**LM35 Precision Centigrade Temperature Sensor [2] and TLC272 Dual Operational Amplifier**

LM35 is a low-cost, precision, semiconductor-based temperature sensor. Its output voltage is linearly proportional to the Centigrade temperature. This complete integrated circuit can continuously measure temperature from –40Co to +110Co with 10mv/1Co.

**Features of LM35:**



The connection diagram for this sensor is shown in Fig(5). Fig(6) shows the connection of TLC272 (Dual Operational Amplifier). The Op-Amp will be used as an interface (SCC) between the temperature sensor and MC9S12DP256 ADC port.

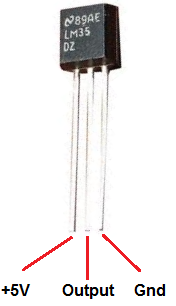
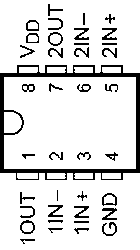
 

Figure 5. TLC272 Figure 6. LM35 Temperature Sensor

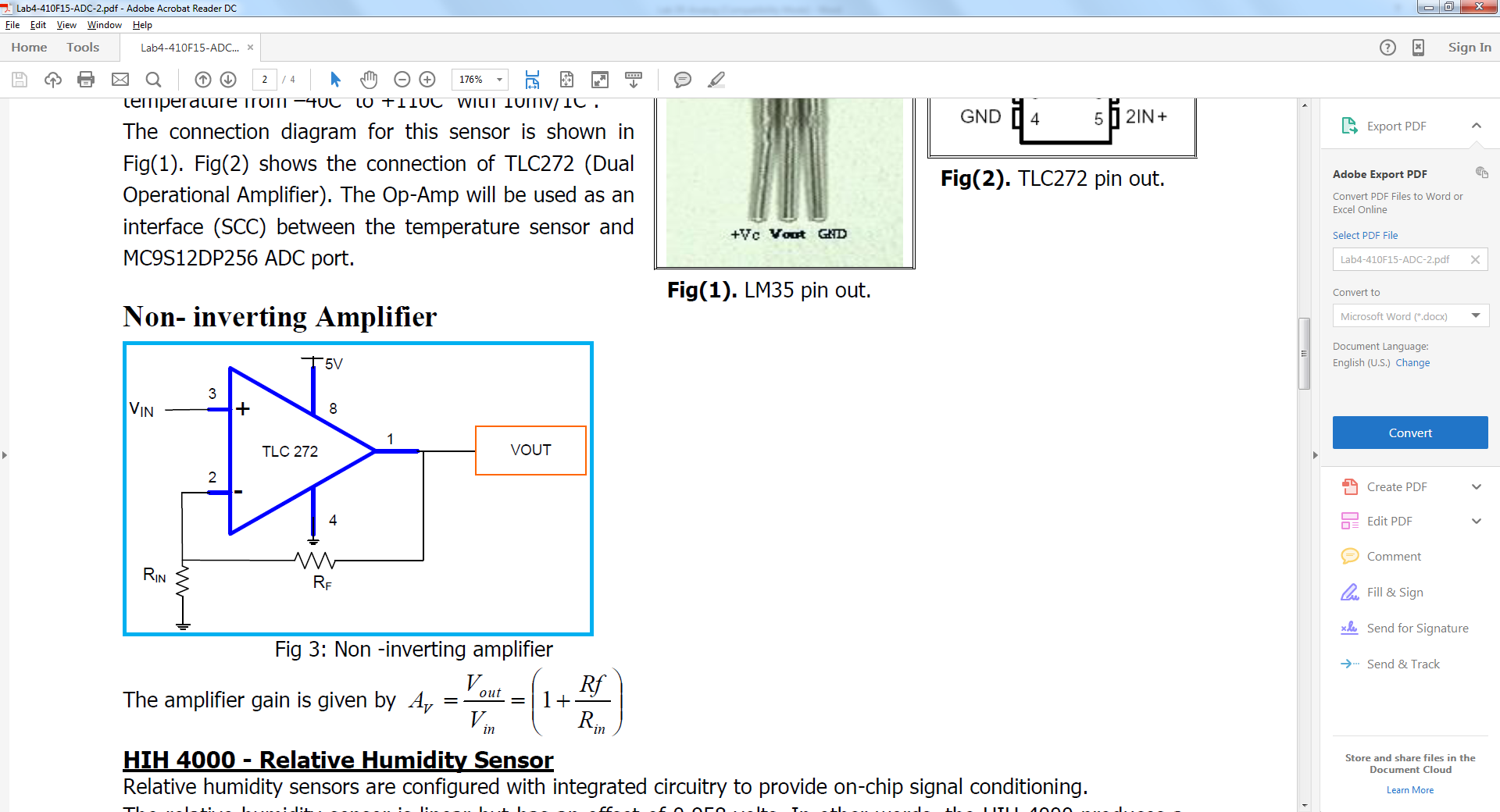
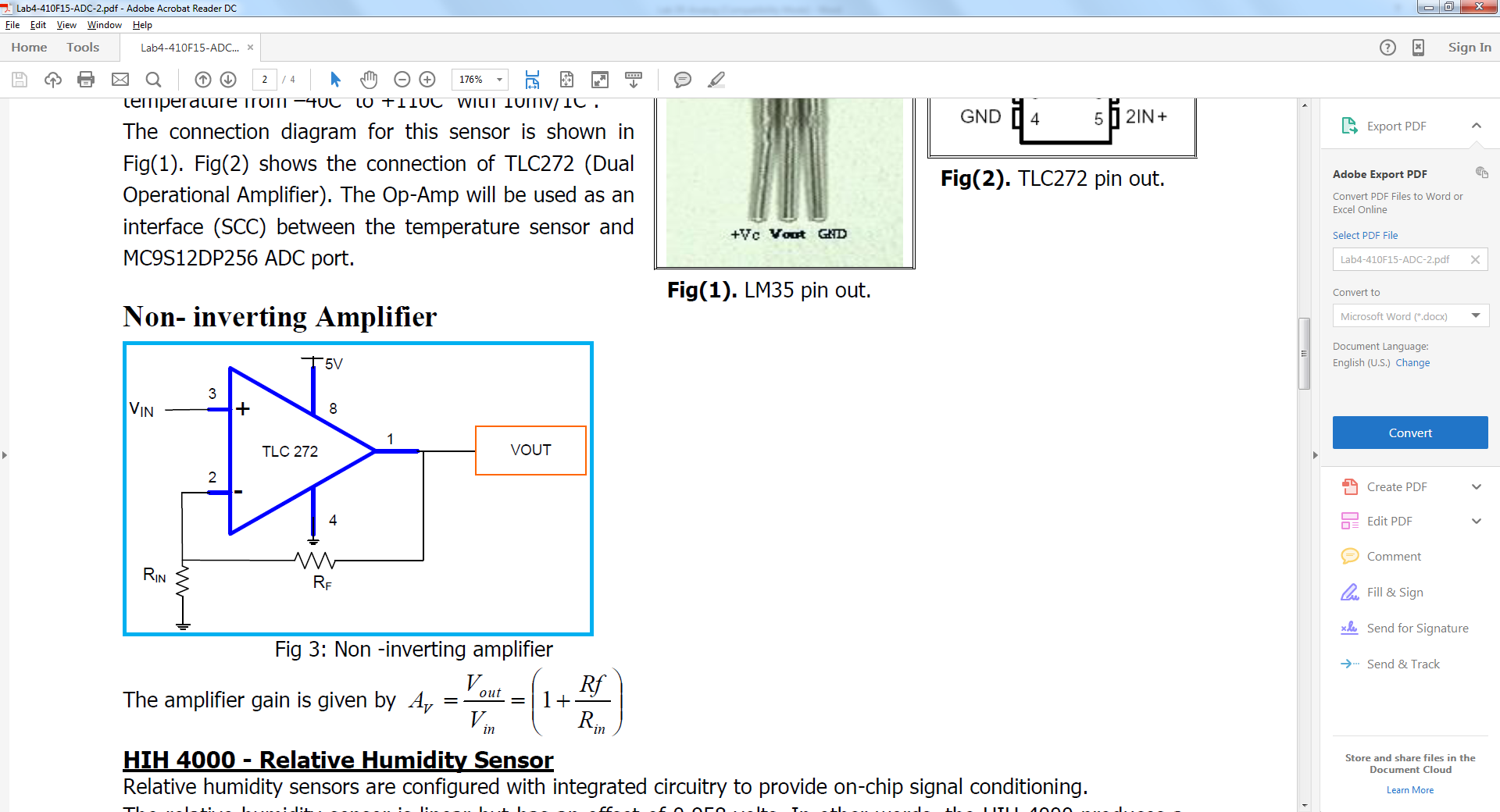


Figure 7. Non-inverting Amplifier

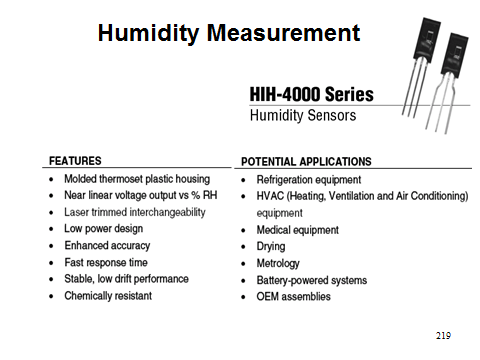
The amplifier gain is given by

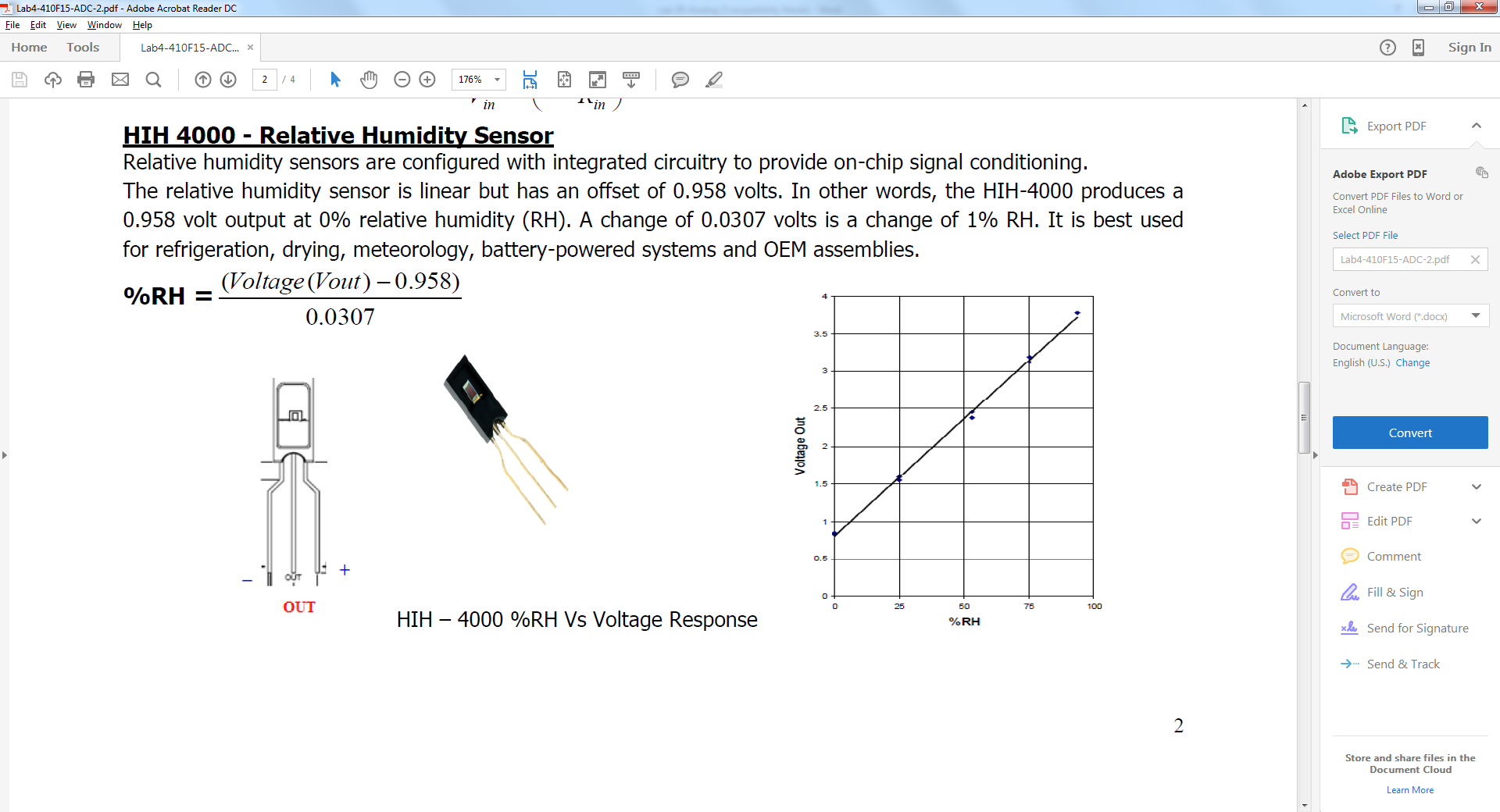


**HIH 4000 - Relative Humidity Sensor [3]**

Relative humidity sensors are configured with integrated circuitry to provide on-chip signal conditioning.

The relative humidity sensor is linear but has an offset of 0.958 volts. In other words, the HIH-4000 produces a 0.958 volt output at 0% relative humidity (RH). A change of 0.0307 volts is a change of 1% RH. It is best used for refrigeration, drying, meteorology, battery-powered systems and OEM assemblies.





**Pre-requisites: Downloading and Installing Required Python Libraries [4]**

Manufacturers of the chip also provides a library that can be used to read values from the ADC Pi Plus. Python requires a library called System Management Bus (SMBus) to communicate with I2C devices. Thus, the first step is to install the library called python-smbus.

To install the library, go to terminal and type the below commands

sudo apt-get update

sudo apt-get install python-smbus

After completing the installation, download the following library from:

<https://github.com/abelectronicsuk/ABElectronics_Python_Libraries/tree/master/ADCPi>

After downloading the library, add the location of the library into PYTHONPATH. PYTHONPATH sets the search path for **importing** python modules.

For instance if the library was Downloaded to Desktop, open terminal and type

export PYTHONPATH=${PYTHONPATH}:~/Desktop/ABElectronics\_Python\_Libraries/ADCPi/

**ADC Pi object Initialization**

ADCPi(smbus, i2c\_address1, i2c\_address1, bit\_rate)

The first argument is the smbus object followed by the two I2C addresses of the ADC chips. The values shown are the default addresses of the ADC board.The forth argument is the sample bit rate you want to use on the ADC chips. Sample rate can be 12, 14, 16 or 18

**Functions offered by ADC Pi Plus Library [4]**

1. Read Voltage value

read\_voltage(channel)

Read the voltage from the selected channel  
**Parameters:** channel - 1 to 8 **Returns:** number as float between 0 and 5.0

1. Set Programmable Gain Amplification

set\_pga(gain)

Set the gain of the PGA on the chip  
**Parameters:** gain - 1, 2, 4, 8 **Returns:** null

1. Set Bit Rate

setBitRate(rate)

Set the sample bit rate of the ADC  
**Parameters:** rate - 12, 14, 16, 18 **Returns:** null

**Options:**  
12 = 12 bit (240SPS max)  
14 = 14 bit (60SPS max)  
16 = 16 bit (15SPS max)  
18 = 18 bit (3.75SPS max)

1. Set Conversion Mode

set\_conversion\_mode(mode)

Set the conversion mode for the ADC  
**Parameters:** mode - 0 = One-shot conversion, 1 = Continuous conversion  
**Returns:** null

**Steps to use ADC Pi Plus Library in Python**

1. Import the Library

from ABE\_ADCPi import ADCPi

1. Import the helper class

from ABE\_helpers import ABEHelpers

1. Get an instance of helper class

i2c\_helper = ABEHelpers()

1. Get access to smbus

bus = i2c\_helper.get\_smbus()

1. Initialize ADCPi object

adc = ADCPi(bus, 0x68, 0x69, 18)

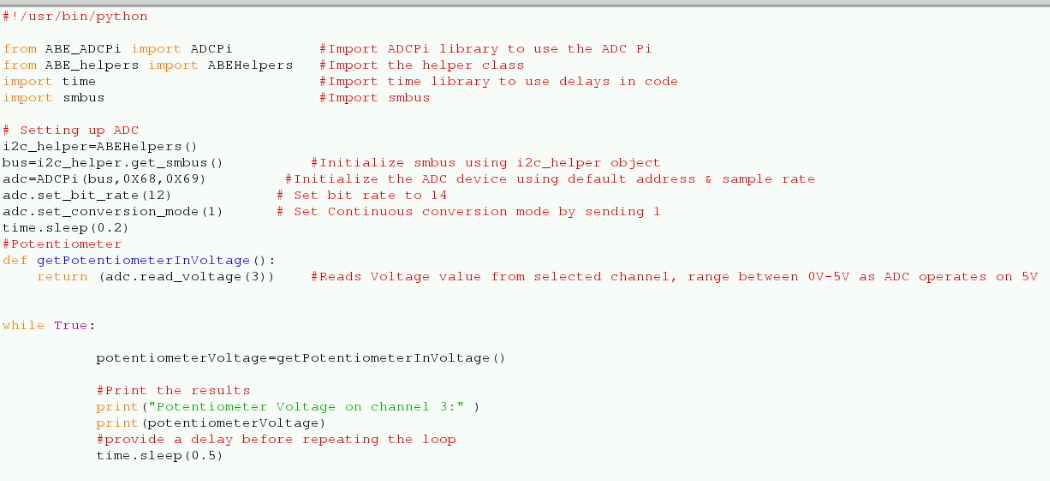
We make chip 1 to be accessible via 0x68 and chip 2 to be accessible via 0x69. Also , we set bit rate to 18.

1. Read values

adc.read\_voltage(1)

**Lab Exercises**

* 1. Write a program to read analog Voltage values from a Potentiometer connected to Channel 3 of ADC. using the bit rate function set the bit rate to 12 bits. Use continuous Conversion mode.



**Output:**



Take 5 different readings for the Potentiometer Analog voltage between 0V to 5V (inclusive). From the experimental analog voltages, calculate the corresponding analog reading from the Potentiometer i.e. the raw analog value from Potentiometer. How will these Analog readings change if the bit rate isset to 18. Explain.

* 1. Write a program to read analog values given by temperature sensor connected at channel 1 and humidity sensor connected at channel 5. Sampling rate must be 18 bits. Please follow the connection diagram (shown in Figure 8) when interfacing sensors to ADC Pi.

Hints:

Humidity Value is given by: (Voltage value- 0.958) / 0.0307

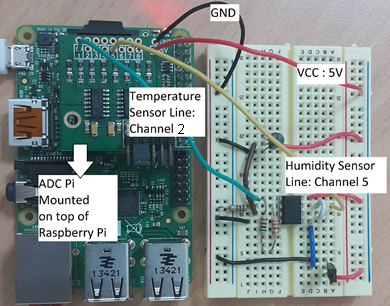
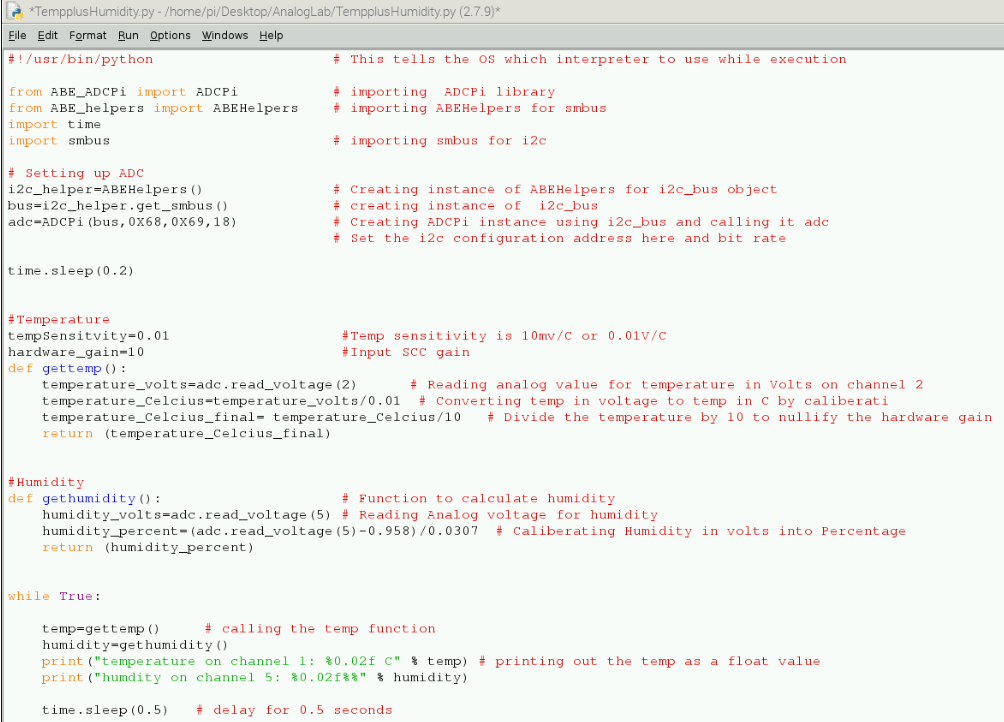
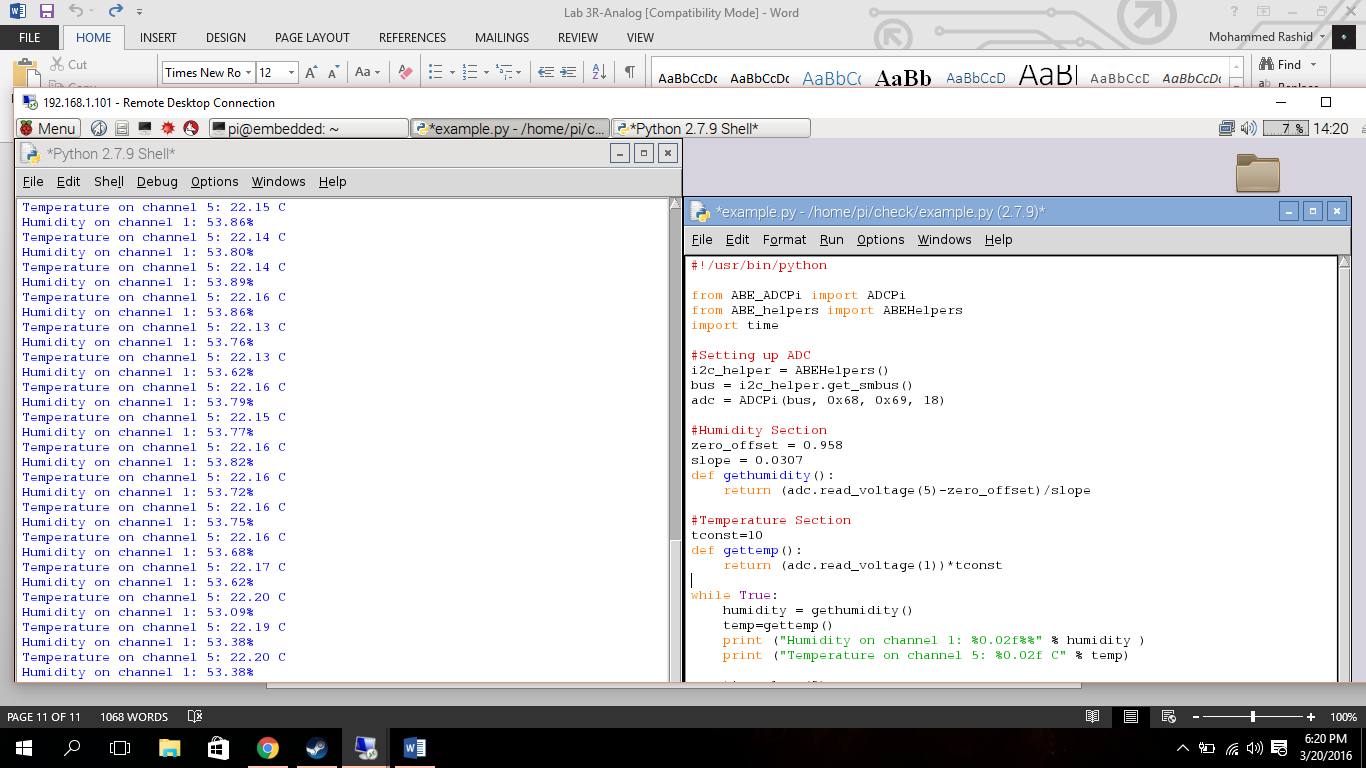


Figure 8. Connection Diagram





**References**

[1] <https://www.abelectronics.co.uk/p/56/ADC-Pi-Plus-Raspberry-Pi-Analogue-to-Digital-converter>

[2] <http://www.ti.com/lit/ds/symlink/lm35.pdf>

[3] <http://www.phanderson.com/hih-4000.pdf>

[4]<https://github.com/abelectronicsuk/ABElectronics_Python_Libraries/tree/master/ADCPi> ]