**INTERNSHIP PROJECT REPORT**

**Development of AD/DA Signal using Raspberry Pi**

Submitted By:

**ANUSKA BOSE**

Reg No - 21BCT0002

Btech in Computer Science and Engineering with Specialization in Internet of Things (SCOPE)



VELLORE INSTITUTE OF TECHNOLOGY,VELLORE

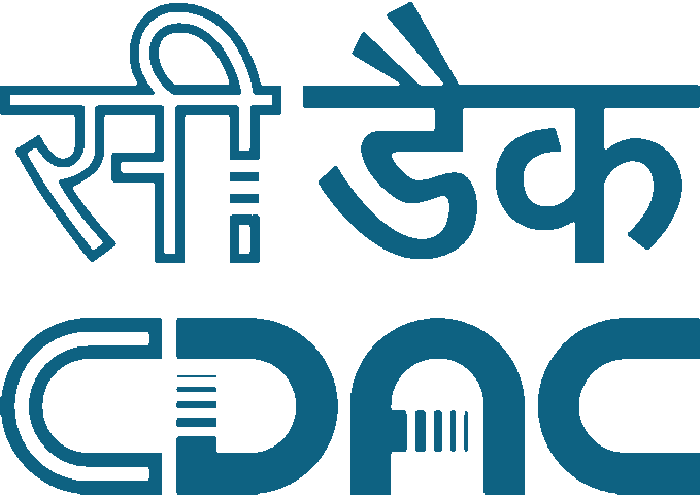
INTERNSHIP DURATION: 24/08/2023 – 05/10/2023

Company Name – **C-DAC KOLKATA**

Under the Guidance of

Dr. Subrata Sarkar

CDAC-KOLKATA



**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Topic** | **Page no.** |
| 1 | Acknowledgment | 1 |
| 2 | Introduction | 2 |
| 3 | Project Methodology | 3-6 |
| 4 | Hardware Description | 7-8 |
| 5 | Software Description | 9 |
| 6 | Code | 10-16 |
| 7 | Inference | 17 |
| 8 | References | 18 |

**ACKNOWLEDGMENT**

I express my deep sense of gratitude with reverence to Centre Head, for his permission and kind support to join the C-DAC, Kolkata for **Industrial Internship 2023.**

My deepest thanks goes to Dr. Subhankar Mukherjee[Project Manager] for his precious time and attention.

I would like to thank Dr. Subrata Sarkar, Joint Director, CDAC, for his guidance, support, and mentorship throughout my internship. His willingness to share his knowledge and expertise has had a profound impact on my professional development.

I would also like to express my appreciation for the infrastructure, resources, and opportunities that CDAC Kolkata provided to enhance my learning and skill development.

In conclusion, I want to convey my deepest gratitude for the chance to intern at CDAC Kolkata. This experience has been instrumental in shaping my career aspirations and has given me a strong foundation to build upon. I look forward to applying the knowledge and skills I have gained during this internship in my future endeavors.

Date: 05.10.23

ANUSKA BOSE

Dept. of SCOPE [VIT VELLORE]

1

**INTRODUCTION**

The Internet of Things (IoT) has revolutionized the way we interact with our surroundings by enabling the seamless integration of physical devices into the digital world.

In this context, this project aims to design and implement an IoT-based High Precision Analog-to-Digital (AD) and Digital-to-Analog (DA) acquisition system, leveraging the capabilities of a Waveshare board and a Raspberry Pi.

**Increasing Need for Precision AD/DA Acquisition**:IoT has found its way into various industries, and the demand for precise and dependable data acquisition and control has surged, the conversion of physical measurements into digital data, as well as the accurate control of actuators through analog outputs, is pivotal.

**The Role of Waveshare Board and Raspberry Pi**

In this project Waveshare's AD/DA expansion board, coupled with the computational power and versatility of the Raspberry Pi, forms a potent combination for building an IoT-based AD/DA system. The Waveshare board offers high-resolution AD/DA converter. The Raspberry Pi, on the other hand, provides a robust platform for data processing.

**The primary objective of the project:**

Understanding the Hardware: Delving into the intricacies of the expansion board, including its technical specifications and capabilities.

Software Integration: Developing software interfaces and applications that leverage the expansion board's AD/DA functionalities.

Testing and Validation: Rigorously testing the accuracy and reliability of AD/DA conversions and software integration.

2

**PROJECT METHODOLOGY**

The efficient implementation of an IoT-based high-precision AD/DA acquisition system, utilizing a Waveshare board and a Raspberry Pi.

**This project involved several well-executed steps:**

1. I made choices when selecting hardware components, considering IoT-compatible devices that would meet the project's technical specifications. This entailed careful evaluation of Raspberry Pi models, ensuring they possessed the computational power and network connectivity required for IoT applications. I also selected high-precision AD/DA boards that aligned with my technical requirements, including resolution and interface compatibility.

2. Select Hardware Components

For this project, the following hardware componentswere used:

* Raspberry Pi 4 Model B 8GB ram
* High-precision AD/DA board (Waveshare)
* Power supply for Raspberry Pi
* LITHIUM BATTERY 2nos(7.4V each 2200mah 40c)
* MicroSD card for Raspberry Pi OS
* Cables and connectors

3.Software Requirements:

* RaspberryPi Imager
* Angry IP scanner
* RealVNC
* PuTTY
* Matlab
* Python 3.10

4. Set Up Raspberry Pi

* Download and install the Raspberry Pi OS on the microSD card.
* Insert the microSD card into the Raspberry Pi and power it up.
* Connect the Raspberry Pi to the internet via Wi-Fi.
* Installation of OS in the SD card and then inserted into raspberry pi board.

3

* Connect the power adaptar to the raspberry pi board and a red light is visible on the board when the switch is turned on.
* With RealVNC, I could access and control my Raspberry Pi,so biostat01 is the name given to the board so including any software or applications related to my AD/DA board. I could interact with the AD/DA functions and monitor data remotely.
* Now register the raspberry pi network to the server then an internet connection is provided to pi and its finally ready to execute the commands
* Open the Raspberry Pi terminal, and enter the configuration interface with the following commands:

sudoraspi-config

Choose Interfacing Options -> SPI -> Yes to enable the SPI interface

4. Install Required Libraries

* Use the Raspberry Pi terminal to install necessary software libraries and tools:
* Install BCM2835 and WiringPi libraries
* Install Python libraries for GPIO control
* Configuration of Raspberry Pi for remote access using VNC

5.Connect the AD/DA Board and run the libraries in the Linux Terminal :

For BCM2835 Library-  cd bcm2835/

make

sudo ./ads1256\_test

WiringPi – cd wiringpi/

Make

Sudo. /ads1256\_test

Python3- cd python3/

Sudo python3 main.py

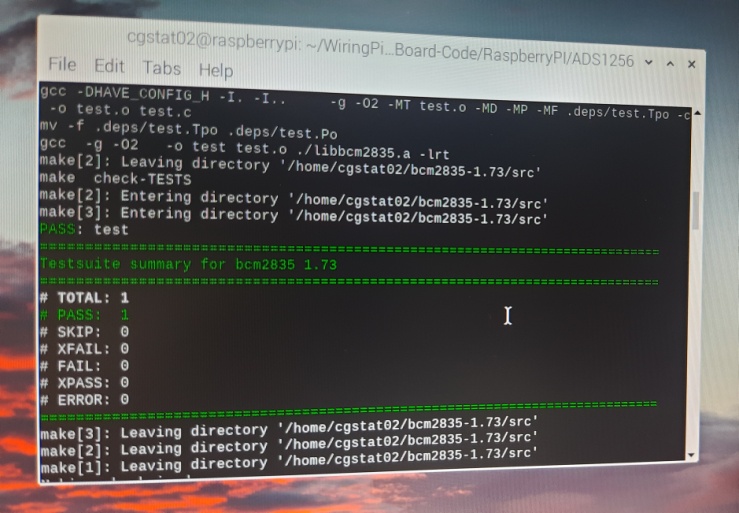
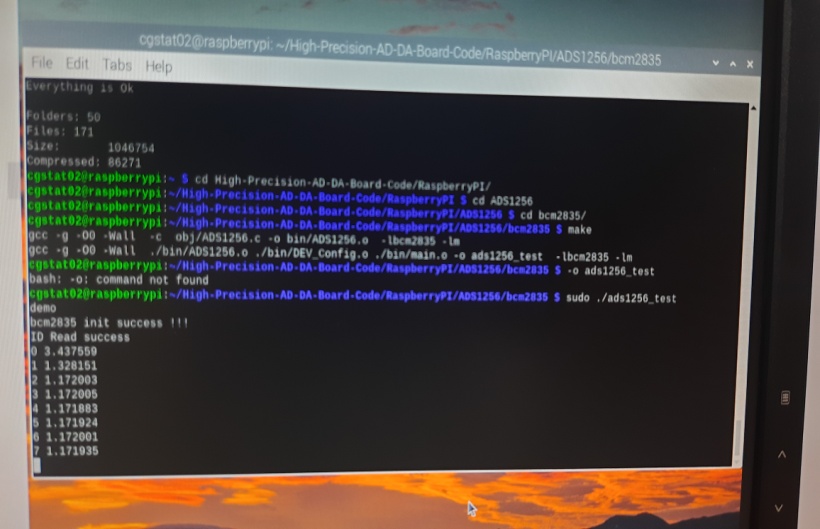
Experiment Phenomenon:

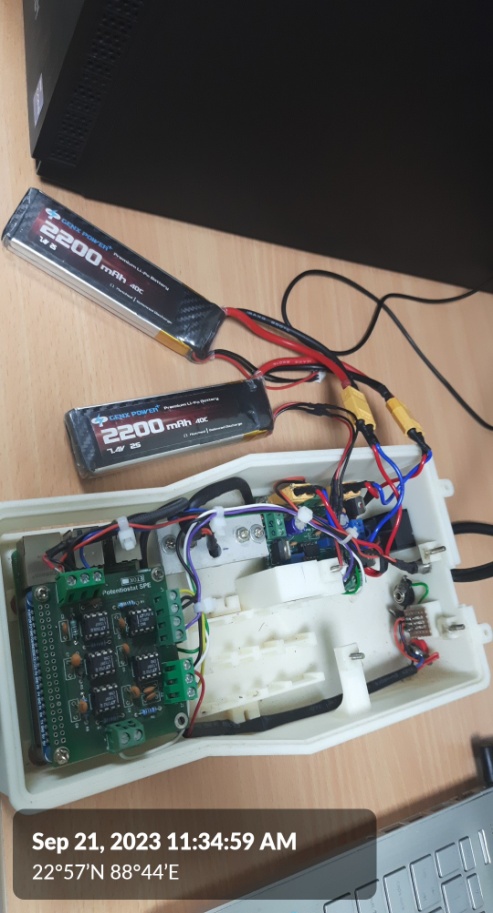
Turn the potentiometer, the AD0 channel voltage will change accordingly.  
Block the photoresistor, and the AD1 channel voltage changes accordingly.

**4**

**AD/DA Demo**

Provide demo: bcm2835, python.

****

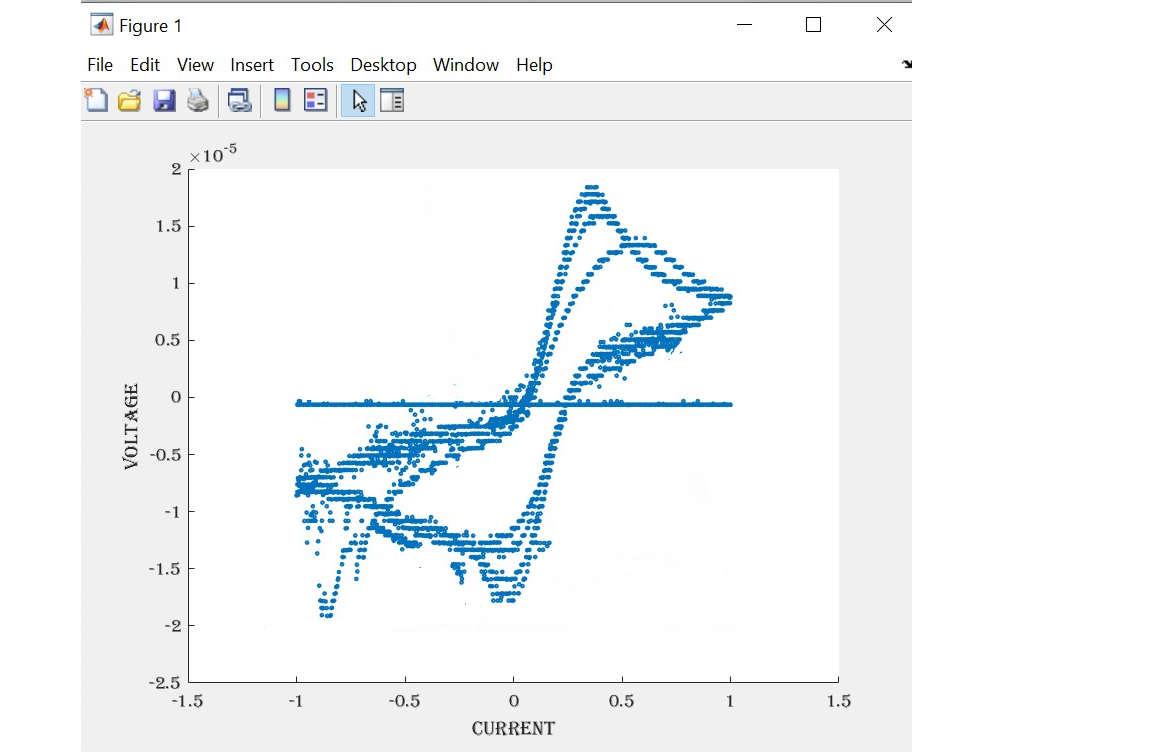
Upon adjusting the potentiometer, the Raspberry Pi will trigger alternating illumination of the two LEDs and generate an ID read success.

****

5

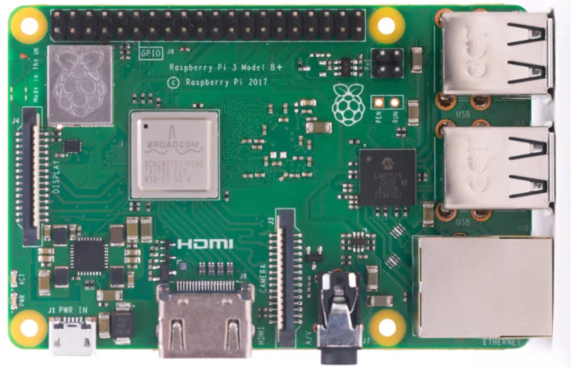
Setup initial voltage to 2.5

The graph is being observed inMatlab with the given set of values and then the data is being noted .



6

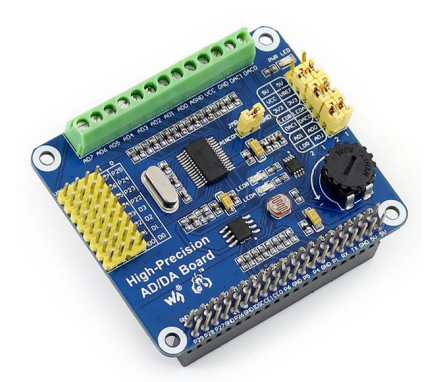
**Hardware Description**



1**.Raspberry Pi 4 Model B with 8GB of RAM**

The Raspberry Pi 4 Model B serves as the central component in the Raspberry Pi AD/DA Expansion Board project, showcasing its ability to connect with external AD to DA converters for precise data acquisition.

|  |  |
| --- | --- |
| Category | Specification |
| Processor | Broadcom BCM2711, Quad-core Cortex-A72 64-bit SoC @ 1.5GHz |
| RAM | 8GB LPDDR4-3200 SDRAM |
| STORAGE | microSD Card Slot (32 GB) |
| USB PORTS | 2 x USB 3.0 Ports , 2 x USB 2.0 Ports |
| VIDEO OUTPUT | 2 x Micro HDMI (up to 4K resolution) |
| AUDIO OUTPUT | 3.5mm Stereo Jack |
| NETWORK | Gigabit Ethernet (RJ-45) |
| WIRELESS CONNECTIVITY | Dual-band 802.11ac Wi-Fi |
| GPIO PINS | 40-pin GPIO Header |
| POWER SUPPLY | 5V/3A via USB-C |
| OPERATING SYSTEM | Raspberry Pi OS (Raspbian) |
| DIMENSIONS | 85.6mm x 56.5mm |
| WIDTH | 46grams(approx) |



2.High-Precision AD/DA Board

The High-Precision AD/DA Board allows you to add high-precision AD/DA functions to the Raspberry Pi.

7

Analog-to-Digital(ADC)

|  |  |
| --- | --- |
| Category | Specification |
| Resolution | Typically 16-bit or 24-bit |
| Input Voltage Range | Varies (e.g., 0-5V, -10V to +10V, etc.) |
| Sampling Rate | Varies (e.g., up to 1 MSPS) |
| Input Channels | Multiple channels (e.g., 4, 8, 16, etc.) |
| Accuracy | High precision (e.g., ±0.1% or better) |
| Interface | SPI or I2C |

Digital-to-Analog (DAC)

|  |  |
| --- | --- |
| Resolution | Typically 16-bit or 24-bit |
| Output Voltage Range | Varies (e.g., 0-5V, -10V to +10V, etc.) |
| Output Channels | Multiple channels (e.g., 1, 2, 4, etc.) |
| Accuracy | High precision (e.g., ±0.1% or better) |
| Output Current | Varies (e.g., mA to A) |
| Interface | SPI or I2C |
| Power Supply | Varies (e.g., 5V or 3.3V) |

3.Lithium batteries

Lithium batteries are rechargeable power sources known for their high energy density. Here are the specifications for two lithium batteries, each with a voltage of 7.4V, a capacity of 2200mAh, and a discharge rate of 40C:

4.XT60 Connectors

I used XT60 connectors to connect a power source, such as a battery pack, to both the AD/DA board and the Raspberry Pi in my project.XT60 connectors provided a secure and stable connection, ensuring a reliable power supply to both the AD/DA board and the Raspberry Pi.

8

**Software Description**

1.RaspberryPi Imager

It simplifies the process of writing Raspberry Pi operating system images to microSD cards, making it easier to set up and configure.

2.Angry IP scanner

It allows users to scan and discover devices on their local network, making it useful for identifying IoT devices connected to the same network as your Raspberry Pi.

3.RealVNC

I ensured that my Raspberry Pi was connected to the internet.I opened RealVNC Viewer and entered the IP address or hostname of my Raspberry Pi in the "VNC Server" field.I was prompted to enter the username and password of my Raspberry Pi when connecting for the first time. I entered the credentials I used to log in to my Raspberry Pi.

Once authenticated, I had remote control of my Raspberry Pi's desktop environment.WithRealVNC, I could access and control my Raspberry Pi, including any software or applications related to my AD/DA board. I could interact with the AD/DA functions and monitor data remotely.

4.PuTTY

PuTTY is a popular terminal emulator and SSH client. It is commonly used to establish secure remote connections to Raspberry Pi devices over SSH (Secure Shell)

5.Matlab

MATLAB is a powerful computational software tool that finds applications in IoT development and data analysis. It can be used for processing and analyzing data collected from IoT sensors connected to a Raspberry Pi.

6.Python 3.10

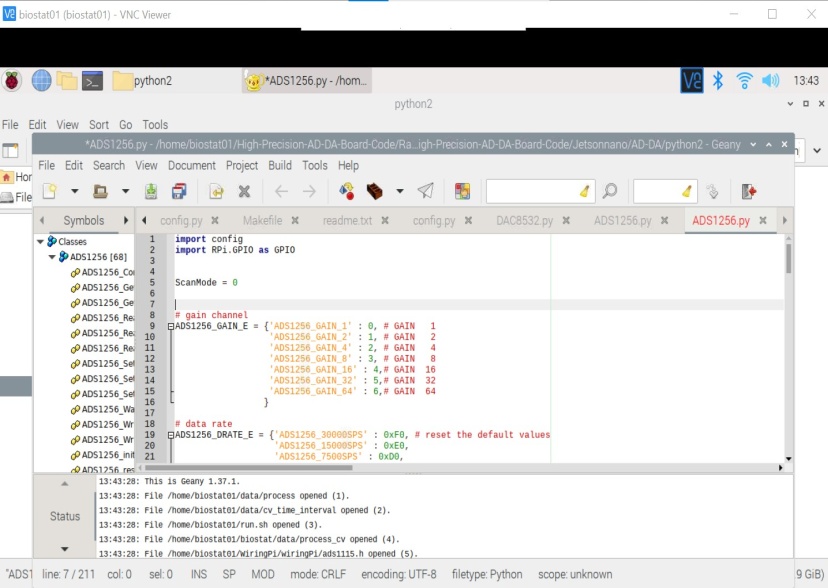
Python is a versatile programming language widely used in IoT development. With Python 3.10 (or any version of Python),I was able to write code and execute for data analysis.

9

**CODE**

1. **AD-DA DEMO**

* ADS1256

This code demonstrates how to read analog input from the ADC channels and convert them into digital values.

import config

import RPi.GPIO as GPIO

ScanMode = 0

# gain channel

ADS1256\_GAIN\_E = {'ADS1256\_GAIN\_1' : 0, # GAIN 1

'ADS1256\_GAIN\_2' : 1, # GAIN 2

'ADS1256\_GAIN\_4' : 2, # GAIN 4

'ADS1256\_GAIN\_8' : 3, # GAIN 8}

def ADS1256\_reset(self):

config.digital\_write(self.rst\_pin, GPIO.HIGH)

config.delay\_ms(20 config.digital\_write(self.rst\_pin, GPIO.LOW)

config.delay\_ms(200)

config.digital\_write(self.rst\_pin, GPIO.HIGH)

def ADS1256\_WriteCmd(self, reg):

config.digital\_write(self.cs\_pin, GPIO.LOW)#cs 0

config.spi\_writebyte([reg])

config.digital\_write(self.cs\_pin, GPIO.HIGH)#cs 1

10

def ADS1256\_GetChannalValue(self, Channel):

if(ScanMode == 0):# 0 Single-ended input 8 channel1 Differential input 4 channel

if(Channel>=8):

return 0

self.ADS1256\_SetChannal(Channel)

self.ADS1256\_WriteCmd(CMD['CMD\_SYNC'])

# config.delay\_ms(10)

self.ADS1256\_WriteCmd(CMD['CMD\_WAKEUP'])

# config.delay\_ms(200)

Value = self.ADS1256\_Read\_ADC\_Data()

else:

if(Channel>=4):

return 0

self.ADS1256\_SetDiffChannal(Channel)

self.ADS1256\_WriteCmd(CMD['CMD\_SYNC'])

# config.delay\_ms(10)

self.ADS1256\_WriteCmd(CMD['CMD\_WAKEUP'])

# config.delay\_ms(10)

Value = self.ADS1256\_Read\_ADC\_Data()

return Value

fori in range(0,8,1):

ADC\_Value[i] = self.ADS1256\_GetChannalValue(i)

returnADC\_Value;

11

**Base File(using C Program)**

#include <stdio.h>

#include <unistd.h>

#include <math.h>

charraw\_file\_name[45],raw\_file\_name\_all[50],raw\_file\_name\_step[50];

FILE \*gp;

chargraph\_string[60];

voidsample\_file\_create(char \*process\_name)

{

charsample\_name[30];

printf("\nPlease enter sample ID or sample name?\t:");

scanf("%s",sample\_name);

sprintf(raw\_file\_name,"%s\_graph\_%s.txt",process\_name,sample\_name);

sprintf(raw\_file\_name\_all,"%s\_graph\_%s\_all.txt",process\_name,sample\_name);

sprintf(raw\_file\_name\_step,"%s\_graph\_%s\_step.txt",process\_name,sample\_name);

}

voidwrite\_values(double V, double I)

{

FILE \*fp=NULL;

fprintf(fp,"%f\t %0.9f\n",V,I)

fclose(fp);

}

{

double value;

12

FILE \*offset\_fp;

offset\_fp=fopen(filename,"r");

if(offset\_fp==NULL) printf("error reading offset\n");

fscanf(offset\_fp,"%lf",&value);

fclose(offset\_fp);

//printf("val=%d\n",value);

return value;

}

voidread\_string(char\* filename, char\* value)

{

//char value[10];

FILE \*offset\_fp;

offset\_fp=fopen(filename,"r");

if(offset\_fp==NULL) printf("error reading offset\n");

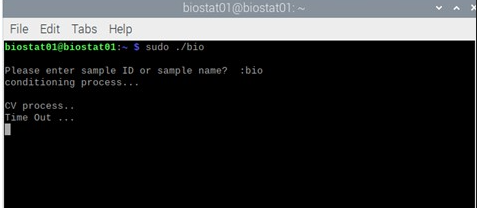
//fscanf(offset\_fp,"%lf",&value);

memset(value,'\0',10);

fgets(value,10,offset\_fp);

fclose(offset\_fp);

//printf("val=%d\n",value);

 //return value;

}

13

1. **CONFIGURATION.py (using Python Program)**

importspidev

import RPi.GPIO as GPIO

import time

# Pin definition

RST\_PIN = 18

CS\_PIN = 22

DRDY\_PIN = 17

# SPI device, bus = 0, device = 0

SPI = spidev.SpiDev(0, 0)

defdigital\_write(pin, value):

GPIO.output(pin, value)

defdigital\_read(pin):

return GPIO.input(DRDY\_PIN)

defdelay\_ms(delaytime):

time.sleep(delaytime / 1000.0)

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(RST\_PIN, GPIO.OUT)

#GPIO.setup(DRDY\_PIN, GPIO.IN)

GPIO.setup(DRDY\_PIN, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

SPI.max\_speed\_hz = 3000000

SPI.mode = 0b01

return 0;

14

**3.For Initialization of Pin Module initialization of BCM2835 library**

#include "DEV\_Config.h"

static void DEV\_GPIOConfig(void)

{

//output

bcm2835\_gpio\_fsel(DEV\_RST\_PIN, BCM2835\_GPIO\_FSEL\_OUTP);

bcm2835\_gpio\_fsel(DEV\_CS\_PIN, BCM2835\_GPIO\_FSEL\_OUTP);

bcm2835\_gpio\_fsel(DEV\_CS1\_PIN, BCM2835\_GPIO\_FSEL\_OUTP);

//intput

bcm2835\_gpio\_fsel(DEV\_DRDY\_PIN,BCM2835\_GPIO\_FSEL\_INPT);

}

UBYTE DEV\_ModuleInit(void)

{

if(!bcm2835\_init()) {

printf("bcm2835 init failed !!! \r\n");

return 1;

} else {

printf("bcm2835 init success !!! \r\n");

}

DEV\_GPIOConfig();

bcm2835\_spi\_begin(); //Start spi interface, set spi pin for the reuse function

bcm2835\_spi\_setBitOrder(BCM2835\_SPI\_BIT\_ORDER\_MSBFIRST); //High first transmission

bcm2835\_spi\_setDataMode(BCM2835\_SPI\_MODE1); //spi mode 0

bcm2835\_spi\_setClockDivider(BCM2835\_SPI\_CLOCK\_DIVIDER\_64); //Frequency

return 0;

}

void DEV\_ModuleExit(void)

{

bcm2835\_spi\_end();

bcm2835\_close();

}

15

**4.MAIN.py (using Python Program)**

#!/usr/bin/python

# -\*- coding:utf-8 -\*-

import time

import ADS1256

import RPi.GPIO as GPIO

try:

ADC = ADS1256.ADS1256()

if (ADC.ADS1256\_init() == -1):

exit()

while(1):

# t=time.time()

ADC\_Value = ADC.ADS1256\_GetAll()

# print " rint " + str((time.time()-t)\*1000)+" mS in reading only channel 0\n "

print ("0 ADC = %.5f"%(ADC\_Value[0]\*5.0/0x7fffff))

print ("1 ADC = %.5f"%(ADC\_Value[1]\*5.0/0x7fffff))

print ("2 ADC = %.5f"%(ADC\_Value[2]\*5.0/0x7fffff))

print ("3 ADC = %.5f"%(ADC\_Value[3]\*5.0/0x7fffff))

print ("4 ADC = %.5f"%(ADC\_Value[4]\*5.0/0x7fffff))

print ("5 ADC = %.5f"%(ADC\_Value[5]\*5.0/0x7fffff))

print ("6 ADC = %.5f"%(ADC\_Value[6]\*5.0/0x7fffff))

print ("7 ADC = %.5f"%(ADC\_Value[7]\*5.0/0x7fffff))

# print "0 ADC = ",(ADC\_Value[0])

# print "1 ADC = ",(ADC\_Value[1])

# print "2 ADC = ",(ADC\_Value[2])

# print "3 ADC = ",(ADC\_Value[3])

# print "4 ADC = ",(ADC\_Value[4])

# print "5 ADC = ",(ADC\_Value[5])

# print "6 ADC = ",(ADC\_Value[6])

# print "7 ADC = ",(ADC\_Value[7])

print ("\33[9A")

except :

GPIO.cleanup()

print "\r\nProgram end "

exit()

16

**Inference**

This project successfully combined Internet of Things (IoT) technology with a Waveshare AD/DA board and Raspberry Pi, a potentiometer, and two LEDs showcasing their seamless compatibility and their potential to deliver high-precision data acquisition. The initial steps involved meticulously connecting the hardware components, ensuring that the AD/DA board was interfaced correctly with the Raspberry Pi.

My goal was to read the analog value from the potentiometer and use it to orchestrate the alternating illumination of the two LEDs.I ensured that the LEDs would alternate their states with precision, synchronizing their illumination with the potentiometer's rotation.I incorporated internet connectivity into the Raspberry Pi, ensuring it could communicate with external systems. The implementation of IoT functionality involved transmitting data, specifically the potentiometer's value .

Rigorous testing and calibration ensured that the LEDs responded precisely to the potentiometer's rotation, delivering the dynamic control that was envisioned.

Ultimately, this project was a testament to the integration of hardware and software, resulting in an IoT-based high-precision AD/DA system using Raspberry Pi.

17

**References**

* <https://www.waveshare.com/wiki/High-Precision_AD/DA_Board>
* <https://pinout.xyz/pinout/high_precision_adda_board>
* <https://www.mouser.com/datasheet/2/744/High_Precision_AD_DA_User_Manual-2487138.pdf>
* <https://how2electronics.com/getting-started-setting-up-raspberry-pi-4-model-b/>
* <https://www.youtube.com/watch?v=zcTzZSP_u4U>
* <https://www.youtube.com/watch?v=JZ1pdVVTMrw>

18