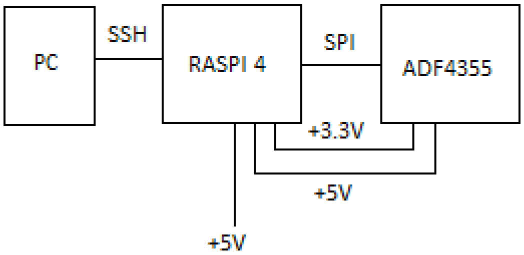
Marcin Czerniewski 12.20.2021

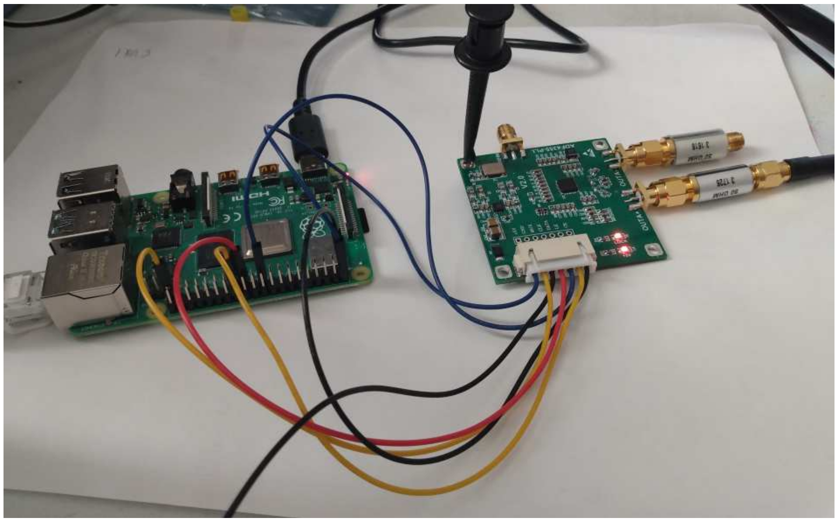
ADF4355 SYNTHESIZER

1. General

The main purpose of his document is to show the programming of the ADF4355 PLL IC and the use of SSH library JSCH written in Java. In order to run this app. Java has to be installed on PC. This is just an educational demo. No claims of any rights are made.
For commercial applications the rights to use the JSch and JPype should be investigated.
The app. runs on Windows 10 Python 3.8.8. It communicates with Raspberry Pi via SSH.

2. Setup





The setup consists of the PC running Python app. which takes frequency input from the user and loads data into ADF4355 thru Raspberry Pi 4.
The ADF4355 is powered from from Raspberry Pi +5V GPIO pin 2
The ADF4355 CE is connected to Raspberry Pi +3.3V GPIO pin 1
The following frequencies were successfully tested:

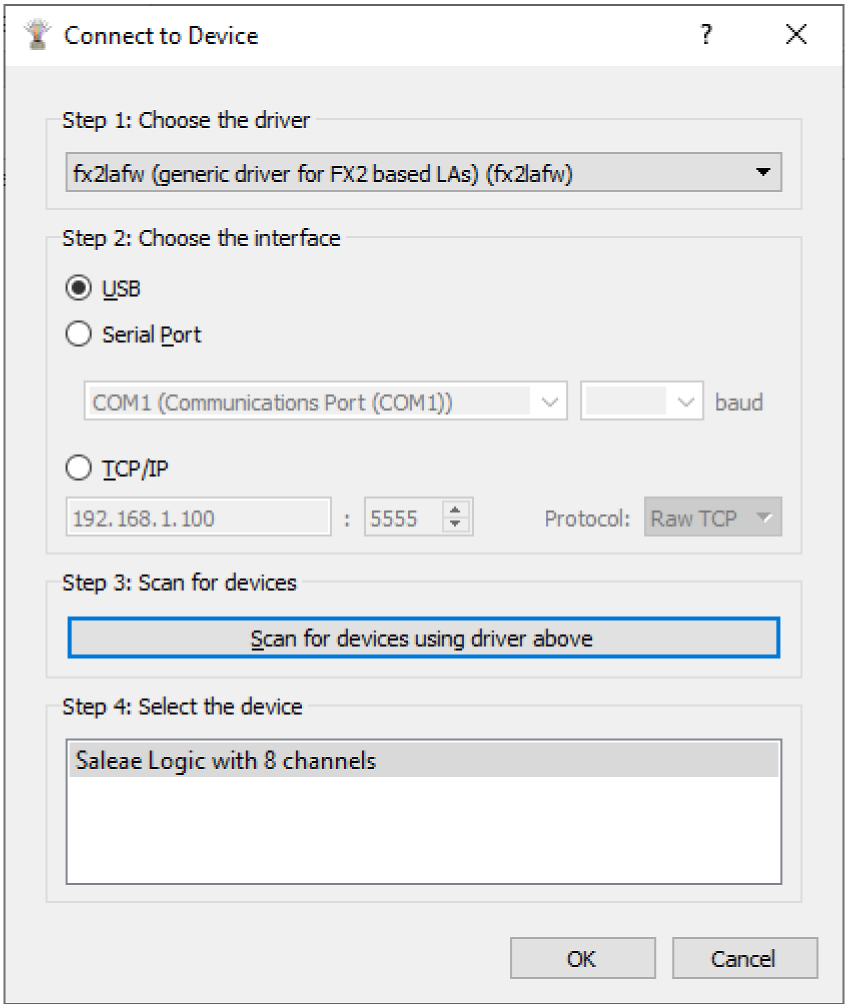
Entered	Measured
MHz	MHz
181.250	181.271
200.050	200.049
350.100	350.098
450.325	450.325
567.075	567.072
643.025	643.021
756.375	756.370

Entered	Measured
MHz	MHz
959.125	959.120
999.999	999.993
1001.825	1001.819
1213.425	1213.418
1305.006	1304.999
1401.033	1401.025

3. Setting up SpiDev

Logic Analyzer used for checking SPI data generated by the Raspberry Pi

Raspi SPI CE=1
Raspi SPI mode=0
Raspi CE active high= False
lsb first=False
loopback= False
RASPI SPI speed=7629



Connect to Device

Step 1: Choose the driver

fx2lafw (generic driver for FX2 based LAs) (fx2lafw)

Step 2: Choose the interface

● USB

○ Serial Port

COM1 (Communications Port (COM1)) baud

○ TCP/IP

192.168.1.100 : 5555 Protocol: Raw TCP

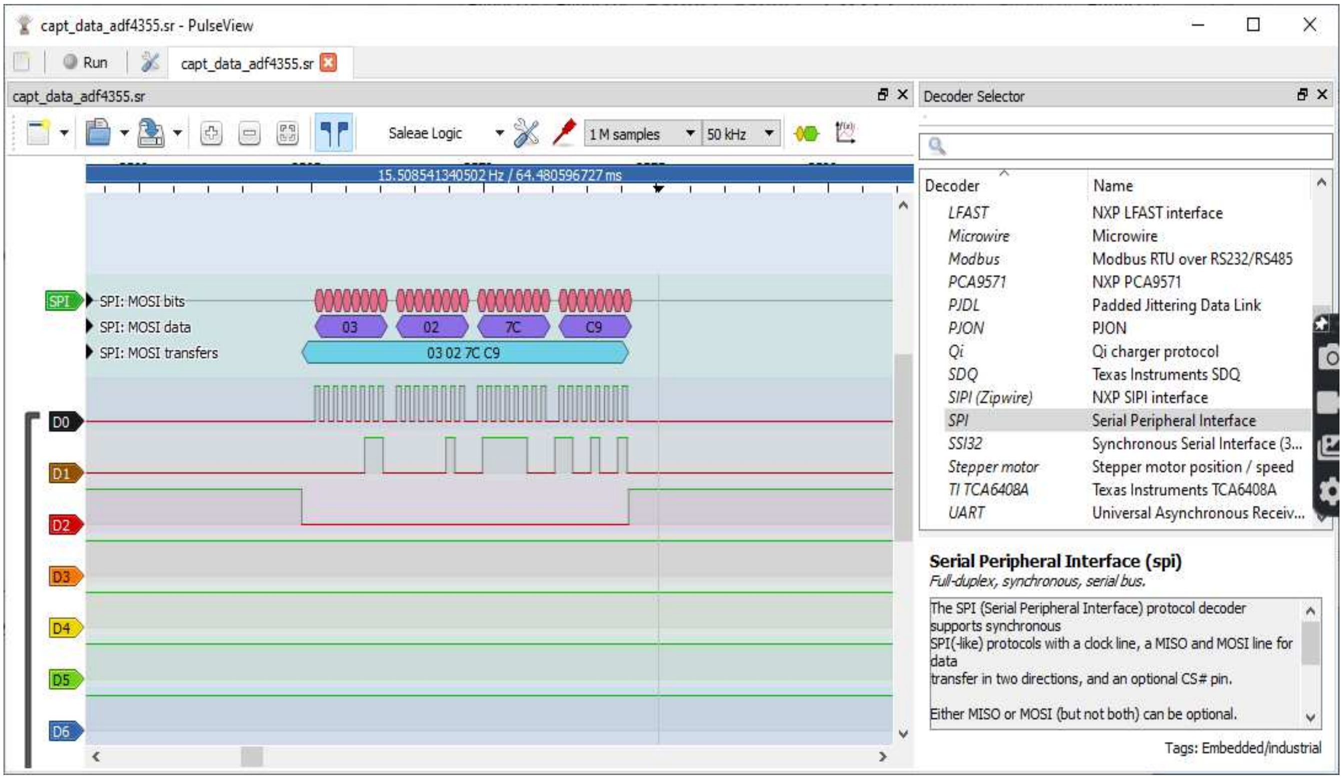
Step 3: Scan for devices

Step 4: Select the device

Saleae Logic with 8 channels

Setting up the Logic Analyzer (Win10)

SPI data generated by Raspberry Pi captured with the Logic Analyzer



Decoder	Name
LFAST	NXP LFAST interface
Microwire	Microwire
Modbus	Modbus RTU over RS232/RS485
PCA9571	NXP PCA9571
PJDL	Padded Jittering Data Link
PJON	PJON
Qi	Qi charger protocol
SDQ	Texas Instruments SDQ
SIPI (Zipwire)	NXP SIPI interface
SPI	Serial Peripheral Interface
SSI32	Synchronous Serial Interface (3...
Stepper motor	Stepper motor position / speed
TI TCA6408A	Texas Instruments TCA6408A
UART	Universal Asynchronous Receiv...

Serial Peripheral Interface (spi)
Full-duplex, synchronous, serial bus.

The SPI (Serial Peripheral Interface) protocol decoder supports synchronous
SPI(-like) protocols with a clock line, a MISO and MOSI line for data
transfer in two directions, and an optional CS# pin.

Either MISO or MOSI (but not both) can be optional.

Tags: Embedded/industrial

4. Data Set that maybe useful for debugging the app.

```
Enter PLL freq in MHz:  1401.033

set PLL to:  1401.033
adc clk= 62
Fpd= 5.0
OutDivider (2^x) x= 2
getSyntSettings RFout= 1401.033
PrescalerInputFrequency= 5604.132
N= 1120.8264
N= 1120.8263999999992
freq err= -3.637979e-06
VCOFreq= 5604.131999999996
self.FRAC1= 13864691
self.FRAC2= 4954
self.MOD1= 16777216
self.MOD2= 16383
OutputDiv= 2 (division 2^2=4)
INT= 1120
R0= 0x204600
R1= 0xd38ef31
R2= 0x4d6bfff2
R3= 0x30000003
R4= 0x30050b84
R5= 0x800025
R6= 0x35402076
R7= 0x120000e7
R8= 0x102d0428
R9= 0x3027cc9
R10= 0xc00fba
R11= 0x61300b
R12= 0x1041c

Regs[]= [2115072, 221835057, 1298923506,
805306371, 805636996, 8388645,
893395062, 301990119, 271385640,
50494665, 12586938, 6369291, 66588]
```

5. Installing SW

PC

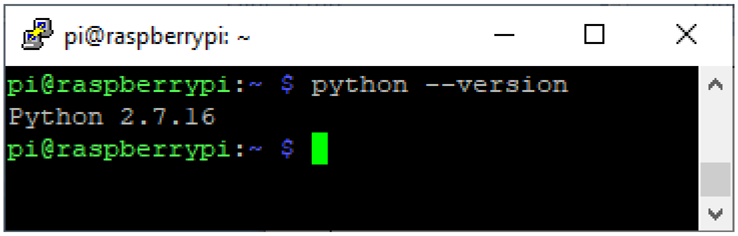
5.1 Install Python 3 or higher and Java JDK 1.8 or higher on your PC

5.2 Copy Java SSH folder to your PC C:/SshJavaJarPy7/
For full information, downloads and usage of Java SSH API example go to www.jcraft.com
Excellent example and description can be found at
https://www.journaldev.com/246/jsch-example-java-ssh-unix-server

5.3 Install JPype 1.3 on your PC
https://pypi.org/project/JPype1/
This is a bridge between Python and Java.
It facilitates access to Java Classes and Jars from Python in a very simple manner.

Raspberry Pi

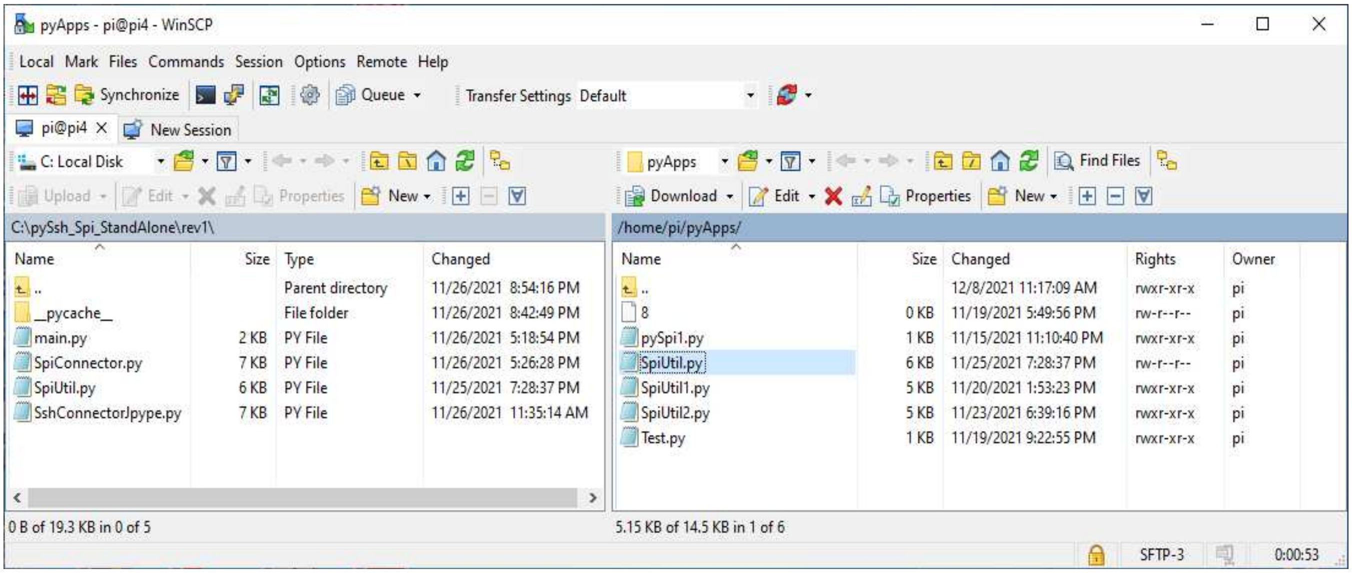
5.4 Check if Python is installed on your Raspberry PI



```
pi@raspberrypi:~ $ python --version
Python 2.7.16
pi@raspberrypi:~ $
```

5.5 Make dir pyApps

5.6 Copy SpiUtil.py to this directory



Transfer Settings Default

C:\pySsh_Spi_StandAlone\rev1\

Name	Size	Type	Changed
..		Parent directory	11/26/2021 8:54:16 PM
__pycache__		File folder	11/26/2021 8:42:49 PM
main.py	2 KB	PY File	11/26/2021 5:18:54 PM
SpiConnector.py	7 KB	PY File	11/26/2021 5:26:28 PM
SpiUtil.py	6 KB	PY File	11/25/2021 7:28:37 PM
SshConnectorJpype.py	7 KB	PY File	11/26/2021 11:35:14 AM

0 B of 19.3 KB in 0 of 5

/home/pi/pyApps/

Name	Size	Changed	Rights	Owner
..		12/8/2021 11:17:09 AM	rwxr-xr-x	pi
8	0 KB	11/19/2021 5:49:56 PM	rw-r--r--	pi
pySpi1.py	1 KB	11/15/2021 11:10:40 PM	rwxr-xr-x	pi
SpiUtil.py	6 KB	11/25/2021 7:28:37 PM	rw-r--r--	pi
SpiUtil1.py	5 KB	11/20/2021 1:53:23 PM	rwxr-xr-x	pi
SpiUtil2.py	5 KB	11/23/2021 6:39:16 PM	rwxr-xr-x	pi
Test.py	1 KB	11/19/2021 9:22:55 PM	rwxr-xr-x	pi

5.15 KB of 14.5 KB in 1 of 6

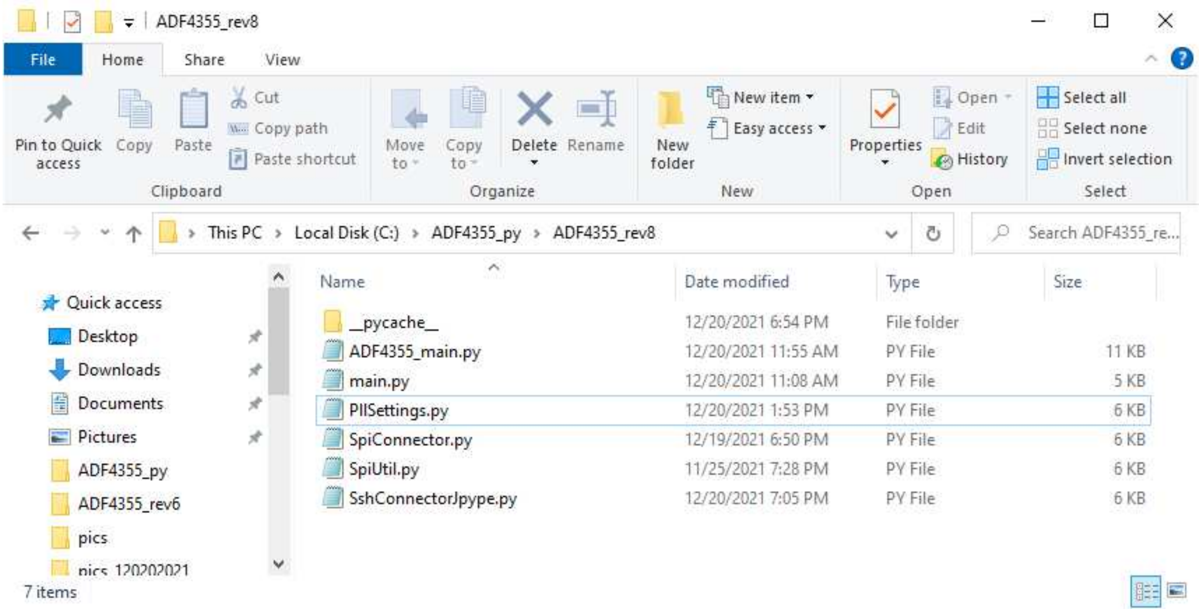
5.7 Install SPIDEV on Raspberry Pi

6. Modify JPype in order to avoid “JVM already started” error

```
_jpype _core.py modified x=0 and x=-1 added
if _jpype.isStarted():
    x=0
    raise OSError('JVM is already started')
    return x
  global _JVM_started
  if _JVM_started:      x=-1
    raise OSError('JVM cannot be restarted')
    return x
```

More elegant modification could be made but since it worked well it was left alone.

7. The app. description

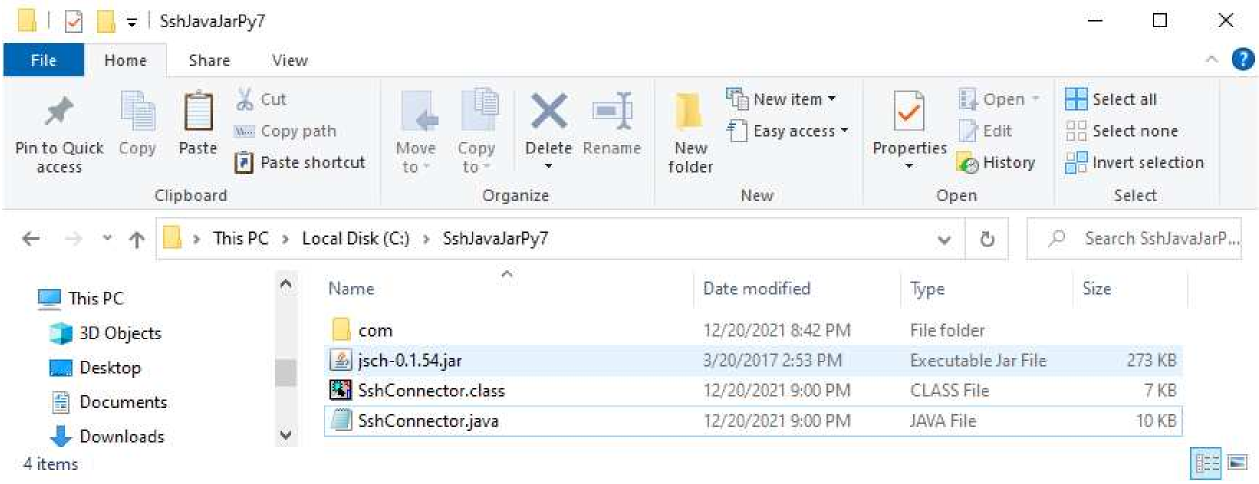


Name	Date modified	Type	Size
__pycache__	12/20/2021 6:54 PM	File folder	
ADF4355_main.py	12/20/2021 11:55 AM	PY File	11 KB
main.py	12/20/2021 11:08 AM	PY File	5 KB
PllSettings.py	12/20/2021 1:53 PM	PY File	6 KB
SpiConnector.py	12/19/2021 6:50 PM	PY File	6 KB
SpiUtil.py	11/25/2021 7:28 PM	PY File	6 KB
SshConnectorJpype.py	12/20/2021 7:05 PM	PY File	6 KB

The app. consists of 5 classes.

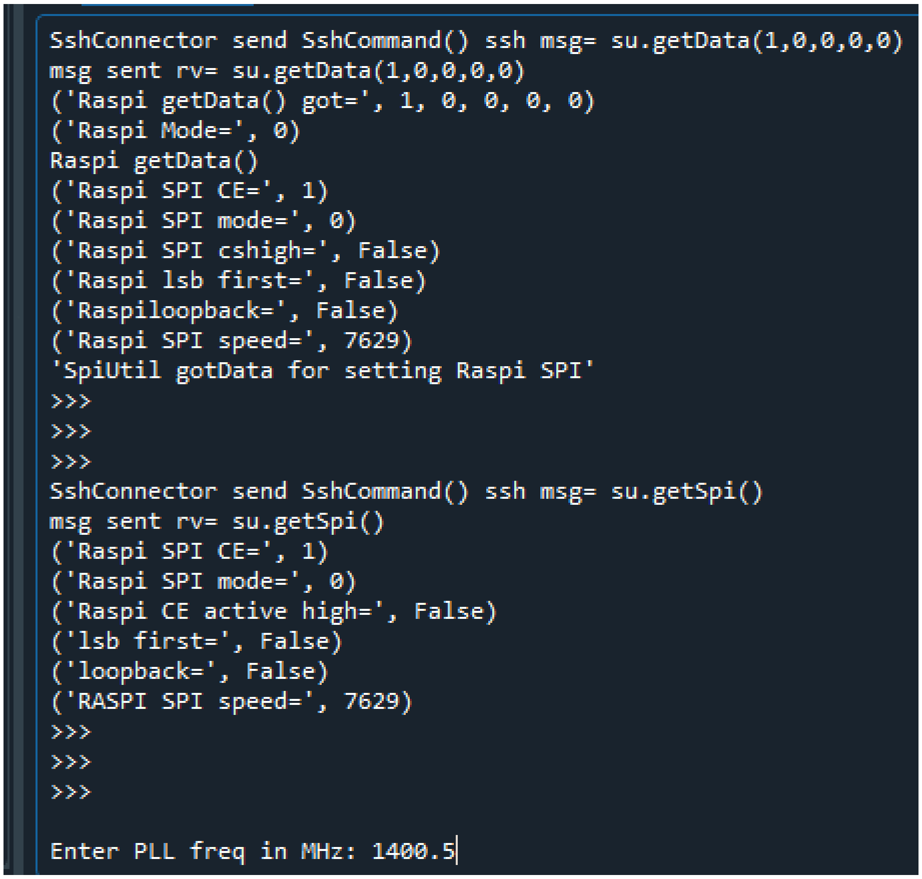
- The main.py inits all other classes
- The PllSettings.py calculates dividers settings
- The SpiConnector.py facilitates data transfer from Raspberry Pi to ADF4355 via SPI
- The SshConnectorJpype.py supports the SSH connection between PC and Raspberry Pi.
- The ADF4355_main.py calculates all registers settings.

There is also another folder that has to be copied to PC C:\ drive.
This is Java SshJavaJarPy7 folder called by SshConnectorJpype.py class.



Name	Date modified	Type	Size
com	12/20/2021 8:42 PM	File folder	
jsch-0.1.54.jar	3/20/2017 2:53 PM	Executable Jar File	273 KB
SshConnector.class	12/20/2021 9:00 PM	CLASS File	7 KB
SshConnector.java	12/20/2021 9:00 PM	JAVA File	10 KB

To run the app. launch main.py and enter frequency in MHz



```
SshConnector send SshCommand() ssh msg= su.getData(1,0,0,0,0)
msg sent rv= su.getData(1,0,0,0,0)
('Raspi getData() got=', 1, 0, 0, 0, 0)
('Raspi Mode=', 0)
Raspi getData()
('Raspi SPI CE=', 1)
('Raspi SPI mode=', 0)
('Raspi SPI cshigh=', False)
('Raspi lsb first=', False)
('Raspiloopback=', False)
('Raspi SPI speed=', 7629)
'SpiUtil gotData for setting Raspi SPI'
>>>
>>>
>>>
SshConnector send SshCommand() ssh msg= su.getSpi()
msg sent rv= su.getSpi()
('Raspi SPI CE=', 1)
('Raspi SPI mode=', 0)
('Raspi CE active high=', False)
('lsb first=', False)
('loopback=', False)
('RASPI SPI speed=', 7629)
>>>
>>>
>>>

Enter PLL freq in MHz: 1400.5
```

8. Final Notes

The app. contains notes, and plenty of print statements that were used for debug .
They were left in place so that the user can see the program flow.

The most interesting part of the code is the use of Java Library in Python application.
The Java Jsch API for SSH works very well in this application and it was quite easy to implement.
In fact it seems to work much better at least for this application than Paramiko and Pexpect.