ÜBUNG 5:

1) IMPLEMENTIERUNG DER A/D-WANDLUNG IN DIE KLASSE SAMPLER

In dieser Übung wird die bisher implementierte Klasse `Sampler` um die Funktionalität einer Analog-Digital-Umsetzung des über das Mikrofon aufgenommenen Signals ergänzt. Nutzen Sie hierfür einen geeigneten internen ADC des ESP32.

a) Fügen Sie die Klassenvariable `adc = ADC(Pin(pinNum), atten=ADC.ATTN_11DB)` der Klasse Sampler hinzu. Dabei wird die Variable `pinNum` auf die von Ihnen benutze Pin-Nummer am ESP32 Modul gesetzt.

b) Implementieren Sie eine Methode `convAD(self, T)` in gleicher Weise, wie die Methode `convDA` umgesetzt ist. Das Argument `T` soll es erlauben eine Aufnahmezeit in Sekunden vorzugeben. Berücksichtigen Sie dies in Ihrer Aufnahmeschleife. Die einzelnen Samples sollen in der bereits vorgegebenen Liste `samples` abgelegt werden.

c) Implementieren Sie eine Methode `startAD(self, T)` in gleicher Weise, wie die Methode `startDA` umgesetzt ist. Berücksichtigen Sie dabei den Parameter `T` für die Aufnahmezeit in Sekunden.

2) TESTEN DER A/D-WANDLUNG

Testen Sie nun Ihre Implementierung. Verwenden Sie zum Beispiel eine App zum Stimmen von Gitarren um sich die Töne der einzelnen Saiten ausgeben zu lassen. Zeichnen Sie Töne auf und geben Sie diese wieder über den Lautsprecher aus. Sie können dann Ihre Tonausgabe wieder mit dem Stimmgerät überprüfen. Verwenden Sie dafür zum Beispiel ein Hauptprogramm in ähnlicher Form wie das folgende:

```
if __name__ == "__main__":
    import gc
    gc.collect()
    print("Freier Speicher: {0}KiB".format(gc.mem_free()/1024))
    fs =6000 #sample freq
    T = 5#record time in sec
    sampler=Sampler(fs)
    for wait in range(3,0,-1):
        print("Sampling starts in {0} seconds...".format(wait))
        time.sleep(1)
    sampler.startAD(T)
    print("Recording...")
    start_ticks = time.ticks_ms()
    sampler.startAD(T)
    while (sampler.conv):pass
    end_ticks = time.ticks_ms()
    print("Sampling after T={0} sec. finished ({1} samples,
{2}KiB).".format(time.ticks_diff(end_ticks,start_ticks)/1000,
len(Sampler.samples),
len(Sampler.samples)*2/1024))
    while True:
        sampler.startDA()
        print("Playback...")
        while (sampler.conv):pass
        print("Done.")
        time.sleep(2)
```

a) Versuchen Sie verschiedene Abtastfrequenzen und beurteilen Sie das Ergebnis. Hinweis: Bei etwa 6 kHz Abtastfrequenz und 2s Aufnahmedauer liegt die Grenze, die mittels MicroPython umgesetzt werden kann. Dies liegt zum einen an der Begrenzung des allokierbaren Arbeitsspeichers und zum anderen an der Verarbeitungsgeschwindigkeit innerhalb der MicroPython-Umgebung. Physikalisch sind deutlich höhere Abtastraten möglich, allerdings muss hierfür der ESP32 hardwarenah in zum Beispiel C programmiert werden. Ein guter Kompromiss sind 4kHz Abtastrate und 4 Sekunden Aufnahmezeit.

3) Abgabe

Laden Sie termingerecht Ihren dokumentierten Python-Code als *.py-Datei auf Moodle hoch.