Guide to running synthesis codes.

1. To connect communication with the Raspberry Pi and set the operation time for a pin. It is necessary to import the following libraries: “RPi.GPIO” and “time”. Also, from the library, “time” import the module “sleep”.

```
1   import RPi.GPIO as GPIO
2   import time
3   from time import sleep
```

2. To activate the electronic components, assign a specific "pin" number for each electronic part and each operation variable. For example, the "peristaltic pump 1" is defined as “pump1 = 13” . The magnetic stirring plate is defined as “stir=12”, and the stirring direction is defined as “lstir=20” and “rstir=21”.

```
5   pump1=13
6   pump2=22
7   pump3=27
8   pump4=17
9   pump5=18
10  stir=12
11  rstir=20
12  lstir=21
```

3. Set the mode to call a pin with its corresponding number with “GPIO.setmode(GPIO.BCM)” and set each pump pin as an output with the command “GPIO.setup(pump1, GPIO.OUT)”.

```
14  GPIO.setmode(GPIO.BCM)
15  GPIO.setup(pump1,GPIO.OUT)
16  GPIO.setup(pump2,GPIO.OUT)
17  GPIO.setup(pump3,GPIO.OUT)
18  GPIO.setup(pump4,GPIO.OUT)
19  GPIO.setup(pump5,GPIO.OUT)
```

4. In the case of the magnetic stirring plate, define its pin as output using the command “GPIO.setup(stir, GPIO.OUT)” and the pins for directions using the command “GPIO.setup(rstir, GPIO.OUT)”. Also, it is necessary to define a frequency variable at which the stirring plate works as “pw_s=GPIO.PWM(stir,500)”. Finally, execute the work's operation using the command “pw_s.start(0)”.

```
21  GPIO.setup(stir,GPIO.OUT)
22  GPIO.setup(rstir,GPIO.OUT)
23  GPIO.setup(lstir,GPIO.OUT)
24  pw_s=GPIO.PWM(stir,500)
25  pw_s.start(0)
```

5. To establish the volume to add to the reaction vial, it is necessary to create variables in each pump defined as “volp1=float(input(“Volume_pump_1(mL): ”))”. Also, a variable to answer Y/N defined as “charge=str(input(“Consider the load time? (Y/N)”))” to consider or rule out the load time during the operation time of each pump. Additionally, a list is created with the flow rate for each pump (flow=[ ]) and another list with the loading time for each pump (tc=[ ]).

```
34   volp1=float(input("Volume_pump_1(mL): "))
35   volp2=float(input("Volume_pump_2(mL): "))
36   volp3=float(input("Volume_pump_3(mL): "))
37   volp4=float(input("Volume_pump_4(mL): "))
38   volp5=float(input("Volume_pump_5(mL): "))
39   charge=str(input("Consider the load time? (Y/N) "))
40   flow=[28,0.24,0.21,0.27,0.28]
41   tc=[1,17,18,18,17]
```

6. To get pumps to work, define for each pump a function as “def Pmp1( ):”. To turn ON the corresponding pump, it is necessary to utilize the command “GPIO.output(pump1,0)”. For easy visualization of the operation of the pumps “print(“Pump1_ON”)” to know that the pump is working and to establish the operation time working use the command “sleep(volp1/flow[0]+tc[0])”. Again, use “print(“Pump1_OFF”)” to know the pump is OFF, and to turn OFF the corresponding pump off, utilize the command “GPIO.output(pump1,1)”. Finally, to establish the end of the pump work, use the command “sleep(1)”. In this case, the work pump is already defined.

```
def Pmp1():
    GPIO.output(pump1,0)
    print("Pump1_ON")
    sleep(volp1/flow[0]+tc[0])
    print("Pump1 _OFF")
    GPIO.output(pump1,1)
    sleep(1)
```

7. In the case of a magnetic stirrer plate, define the operation “def sti1():”. To turn ON the magnetic stirred plate use the command “GPIO.output(rstir,0)”. To know if the stir is working, use “print(“Stir_ON”))”. Also, to set the stirring velocity, use the command “pw_s.ChangeDutyCycle(65)”. The range of the velocity is from 0 to 100, select 65 to establish a medium velocity, enough for a reaction work. To establish the operation time of the stirring plate, use the command “sleep(5)”. This time will change according to the mixture time between reagents before adding another reagent or the reaction time with only stirring.

```
def sti1():
    GPIO.output(rstir,0)
    print("Stir_ON ...")
    pw_s.ChangeDutyCycle(65)
    sleep(5)
    print("Stir_OFF")
    GPIO.output(rstir,1)
    sleep(1)
```

8. If the variable charge is set as “Y” with a conditional sentence as “if charge == "Y":" establish the operation time considering the load time in addition to the volume/flow ratio.

```
44  if charge == "Y":
45
46      def Pmp1():
47          GPIO.output(pump1,0)
48          print("Pump1_ON")
49          sleep(volp1/flow[0]+tc[0])
50          print("Pump1 _OFF")
51          GPIO.output(pump1,1)
52          sleep(1)
53      def Pmp2():
54          GPIO.output(pump2,0)
55          print("Pump2_ON")
56          sleep(volp2/flow[1]+tc[1])
57          print("Pump2_OFF")
58          GPIO.output(pump2,1)
59          sleep(1)
60      def sti1():
61          GPIO.output(rstir,0)
62          print("Stir_ON ...")
63          pw_s.ChangeDutyCycle(65)
64          sleep(5)
65          print("Stir_OFF")
66          GPIO.output(rstir,1)
67          sleep(1)
68      def Pmp3():
69          GPIO.output(pump3,0)
70          print("Pump3_ON")
71          sleep(volp3/flow[2]+tc[2])
72          print("Pump3_OFF")
73          GPIO.output(pump3,1)
74          sleep(1)
75      def Pmp4():
76          GPIO.output(pump4,0)
77          print("Pump4_ON")
78          sleep(volp4/flow[3]+tc[3])
79          print("Pump4_OFF")
80          GPIO.output(pump4,1)
81          sleep(1)
82      def sti2():
83          GPIO.output(rstir,0)
84          print("Stir_ON ...")
85          pw_s.ChangeDutyCycle(65)
86          sleep(1500)
87          GPIO.output(rstir,1)
88          print("Stir_OFF  ")
89          sleep(1)
90      def Pmp5():
91          GPIO.output(pump5,0)
92          print("Pump5_ON")
93          sleep(volp5/flow[4]+tc[4])
94          print("Pump5_OFF")
95          GPIO.output(pump5,1)
96          sleep(1)
```

9. Otherwise, if the variable charge is set as “N” use the command “ elif charge == “N”: ” to establish the operation time only considering the volume/flow ratio.

```
elif charge == "N":
    def Pmp1():
        GPIO.output(pump1,0)
        print("Pump1_ON")
        sleep(volp1/flow[0])
        print("Pump1_OFF")
        GPIO.output(pump1,1)
        sleep(1)
    def Pmp2():
        GPIO.output(pump2,0)
        print("Pump2_ON")
        sleep(volp2/flow[1])
        print("Pump2_OFF")
        GPIO.output(pump2,1)
        sleep(1)
    def sti1():
        GPIO.output(rstir,0)
        print("Stir_ON ...")
        pw_s.ChangeDutyCycle(65)
        sleep(5)
        GPIO.output(rstir,1)
        print("Stir_OFF")
        sleep(1)
    def Pmp3():
        GPIO.output(pump3,0)
        print("Pump3_ON")
        sleep(volp3/flow[2])
        print("Pump3_OFF")
        GPIO.output(pump3,1)
        sleep(1)
    def Pmp4():
        GPIO.output(pump4,0)
        print("Pump4_ON")
        sleep(volp4/flow[3])
        print("Pump4_OFF")
        GPIO.output(pump4,1)
        sleep(1)
    def Pmp5():
        GPIO.output(pump5,0)
        print("Pump5_ON")
        sleep(volp5/flow[4])
        print("Pump5_OFF")
        GPIO.output(pump5,1)
        sleep(1)
    def sti2():
        GPIO.output(rstir,0)
        print("Stir_ON ...")
        pw_s.ChangeDutyCycle(65)
        sleep(1500)
        GPIO.output(rstir,1)
        print("Stir_OFF ")
        sleep(1)
```

10. In the last case, when the charge does not be “Y” or “N”, use the command “else: ” and use “print(“Only Y or N”)” to advertise that have only two options, and put again to define the variable “charge=str(input(“Consider the load time? (Y/N) “))”.

```
else:
    print("Only Y or N")
    charge = str(input("Consider the load time? (Y/N) "))
```

11. Call the functions in order of appears, first appearance pump 2 “Pmp2()”, second appearance pump 3 “Pmp3()”, stirring “sti1()”, third appearance pump 3 “Pmp3()”, stir continues “sti1()”, fourth appearance pump 4 “Pmp4()”, stir continues “stir1()”, fifth and last appearance pump 5 “Pmp5()” and the stirring continues until the reaction time has elapsed “sti2()”. Finally, it is important to include the command “GPIO.cleanup()” to clean up all the pins that have been used.

```
155   Pmp2()
156   Pmp3()
157   sti1()
158   Pmp1()
159   sti1()
160   Pmp4()
161   sti1()
162   Pmp5()
163   sti2()
164
165   GPIO.cleanup()
```