**Index code BApp**

# GENERAL

The code is divided in 3 parts:

1. Importation of modules and definition of GPIOs
2. The functions
3. The architecture of the GUI

The code needs to start by importation of several modules as guizero (GUI), picamera (control of the Raspberry Pi camera v2), time and datetime.

Functions of guizero used in this code:

* Box
* CheckBox
* ComboBox
* PushButton
* Slider
* Text
* TextBox
* ###.tk.configure(): design of the GUI with advanced features from tkinter.

For more information about the architecture of the GUI,

refer to <https://lawsie.github.io/guizero/about/>

Then the electronic parts (LED, motors) are attributed to one or several GPIOs. Each motor is connected to 4 GPIOs. The halfstep\_seq is the activation sequence of the GPIOs controlling the motors. It determines the direction of rotation of the motor: the motor turns anticlockwise, if the sequence is read in the reverse way, the motor turns clockwise.

PWM allows to vary the brightness intensity of LEDs by modulating the voltage they receive. Here between 0% (light off) and 100%.

The different lists are used to register coordinates of the stage current position and fixed points.

IF\_CLOSED():

* Open a window asking if the user wants to close the application.
* Close the camera preview, light off the LEDs and close the application.

# LIGHTS

BF\_ON():

* Light on the blue LED at the intensity chosen by the intensity slider (between 1 and 100).

INTENSITY\_LIGHT\_BF():

* Modify the intensity of the blue LED
* Pwm = % of the voltage intensity the LED receives:
  + 0: the LED is off
  + 100: the LED is at full intensity

BF\_OFF():

* Light off the blue LED, put the voltage received by the LED to 0.

The previous functions are also used to control the other motors.

For: see:

BLUE\_ON(), RED\_ON() BF\_ON()

INTENSITY\_LIGHT\_BLUE(), INTENSITY\_LIGHT\_RED() INTENSITY\_LIGHT\_BF()

BLUE\_OFF(), RED\_OFF() BF\_OFF()

# CAMERA

PREVIEW\_CAMERA():

* Start the camera preview.
* The size of the window and the position in the screen can be modified, as well as the orientation of the image recorded by the camera.
* Change the color of the pushbutton when the camera is on.

CAPTURE\_CAMERA():

* Take a picture.
* Save it on the Desktop with the name chosen in the white box and the date.

BRIGHTNESS\_CAMERA():

* Modify the brightness of the image (post-processing).

CONTRAST\_CAMERA():

* Modify the contrast of the image (post-processing).

OFF\_CAMERA():

* Close the camera preview.
* Reset the background of the preview pushbutton and the choice of resolution.

RESOLUTION\_CAMERA():

* Apply the highest framerate for the RPi camera.
* Apply the chosen resolution in the combobox.
* Modify the background of the pushbutton when the new resolution is set.

# MOTORS

X\_PINS():

* Set the GPIO used to control the X axis.

MOVE\_XForw():

* Define the direction of the motor rotation (anticlockwise) and the time between each range of half-steps.

MOVE\_XBack():

* Define the direction of the motor rotation (clockwise) and the time between each range of half-steps.

RESET\_MOTORX():

* Reset the cycle of steps to 0 and switch off the GPIOs controlling the motor.

XFORWARD():

* Control of the X motor, it will carry out the number of steps written in the white box in the anti-clockwise direction.
* The number of steps is kept in the x2 list. The sum of all steps give the position of the stage relative to the 0 (chosen by the user). The position is written between the pushbutton arrow used to move the motor.
* The speed of the rotation is controlled by the choice in the speed combobox. The time between each range of halfstep is modified. Fast is the maximal speed possible with the configuration where the motors are controlled by the RPi and not by an Arduino.
* After the movement, the GPIOs controlling the motor are set back to 0, this way the motor will start again a complete sequence of steps and not starting in the middle. This allow to have more precise movements.

XBACKWARD():

* Same as XFORWARD() but in the clockwise direction.
* The number of steps kept in the x2 list will be negative.

The previous functions are also used to control the other motors.

For: see:

Y\_PINS(), Z\_PINS() X\_PINS()

MOVE\_YForw(), MOVE\_ZDown() MOVE\_XForw(),

MOVE\_YBack(), MOVE\_ZUp() MOVE\_XBack(),

RESET\_MOTORY(), RESET\_MOTORZ() RESET\_MOTORX()

YFORWARD(), ZDOWN() XFORWARD()

YBACKWARD(), ZUP() XBACKWARD()

SET\_0():

* When the motors are moving, their steps are saved in a list and the sum of the numbers give the position of the stage. Here x2 and y2 lists are emptied. The position of the stage is chosen as the 0 point (the origin).

GOTO\_ZERO():

* X and Y motors do the number of steps required to go back to the position chosen as the origin 0.
* First the X motor will reach the 0 and then the Y.
* Both lists (x2 and y2) are emptied and the GPIOs controlling the X and Y motors are switched off.

SET\_FOCUS():

* Same as SET\_0() but for the Z axis.
* The FOCUS point is the lowest point that the objective can reach.

GOTO\_FOCUS():

* The motor Z does the number of steps required to go back to the position chosen as FOCUS.
* It is assumed that the FOCUS point is the lowest point defined in Z axis. When the pushbutton is clicked the objective will move away from the sample.
* The z2 list is emptied as the Z axis is back to 0 and the GPIOs are switched off.

SAVE\_POSITION\_A():

* The program allows to save positions on the stage to come back to a ROI of the sample easily.
* Before saving any new position in the lists, those are emptied.
* The positions are calculated function to the distance to 0. The list axy gives the coordinates of the point a[X,Y]. Those coordinates are written next to the A pushbutton.
* When a position is chosen, the pushbutton changes color.
* To gain some time during the experiments, the points are saved as relative positions between each other. The A, B, C and D lists correspond to those relative coordinates.

This way there is a path: the stage is at the 0 position, then the A position, then B, C and D and finally goes back to 0. So, it is important to first register the A position then B, C and D.

Those lists contains different information about the point: A[X,Y,Z1,Z2].

SAVE\_POSITION\_AZ1():

* The program allows to do stacks. For this, it is necessary to save the limits in z axis.
* In the third position of the A list is registered the Az1 position. This is the lowest point wanted for the stack.
* Before saving the new position, the 3rd item is cleared.
* The position chosen is written next to the pushbutton.

SAVE\_POSITION\_AZ2():

* In the fourth position of the A list is registered the Az2 position. This is the highest point wanted for the stack.
* Before saving the new position, the 4th item is cleared.
* The position chosen is written next to the pushbutton.

This is important to save the different coordinates sequentially:

1. X and Y
2. Z1
3. Z2

GOTOA():

* The stage is going back to the A position. It needs to be at the origin position.
* First the X position is set, then the Y.
* x2 and y2 lists are emptied and completed with the A coordinates.
* The GPIOs controlling both motors are switched off.

The others positions functions are based on the previous ones.

For: see:

SAVE\_POSITION\_B(), SAVE\_POSITION\_C(),

SAVE\_POSITION\_D() SAVE\_POSITION\_A()

SAVE\_POSITION\_BZ1(), SAVE\_POSITION\_CZ1(),

SAVE\_POSITION\_DZ1() SAVE\_POSITION\_AZ1()

SAVE\_POSITION\_BZ2(), SAVE\_POSITION\_CZ2(),

SAVE\_POSITION\_DZ2() SAVE\_POSITION\_AZ2()

GOTOB(), GOTOC(), GOTOD() GOTOA()

# EXPERIMENT

OPEN\_EXPERIMENT\_WINDOW():

* When the EXPERIMENT pushbutton is clicked, the experiment window shows up.
* In this window, the user can choose the name of the captured pictures during the experiment. The date and time will be added automatically.
* [Time] correspond to how many sets of pictures the user wants to do. Total time is the duration of the total experiment. Both should be in minutes.
* The user can also choose which illumination should be used during the experiment. Brightfield and/or Blue LED or Red LED.

GET\_FOLDER():

* Browse the target folder where the pictures of the experiment will be saved.

START\_EXPERIMENT():

* Ask if the settings chosen are right.
* If the answer is YES: the experiment starts.
* If NO: after closing the popup window, the user can modify the settings.

EXPERIMENT():

* First the time between 2 sets of pictures is calculated as delta\_t.
* PIC correspond to how many sets of pictures will be taken during the experiment, including one at the beginning of the experiment (t=0) and at the end.
* The stage goes to the 0 origin and to the FOCUS point.
* Then a loop starts:

1. The stage is going to the next point of interest and take the pictures: first for the point A, then B, C and D. See GOTO\_NEXT() for more details.
2. Goes back to the 0 origin and to the FOCUS point.
3. Wait during the duration of delta\_t.

* At the end of the experiment, a pop-up window will appear to signal that it is done.

GOTO\_NEXT():

* Coordinates groups the lists of relative positions A,B,C,D. If a point was saved for those positions, the try loop is occurring:

1. The Z axis goes to the FOCUS point.
2. The stage moves to reach the position xy.
3. Depending the choice of illumination, the corresponding LED lights on. Then wait 1s for the stabilization of the light.
4. The z axis moves to reach the z1 position, the (first) picture is taken.
5. If a z2 position is set, several pictures will be taken to make a stack. It depends on how much is big the interval between z1 and z2 and the number of “steps stacks” chosen.
6. The lists are modified to indicate the position of the stage to the program.
7. If a following position has been defined, the loop starts again identically for those positions.

* The pictures are saved in the folder chosen previously with the following name format:

“experiment\_position\_date\_time\_z” e.g.: “protoplasts\_A\_20200822\_1605\_90”

* Once all defined positions have been explored and pictured, the lights are turned off as well as the GPIOs controlling the motors.

STEPS\_STACK():

* Print the number of z steps for stacks.
* This function is here to allow the use of the slider widget in the GUI.

MOVE\_Z\_STACKS():

* Move the Z motor up according how many steps have been chosen between 2 pictures for stacks.