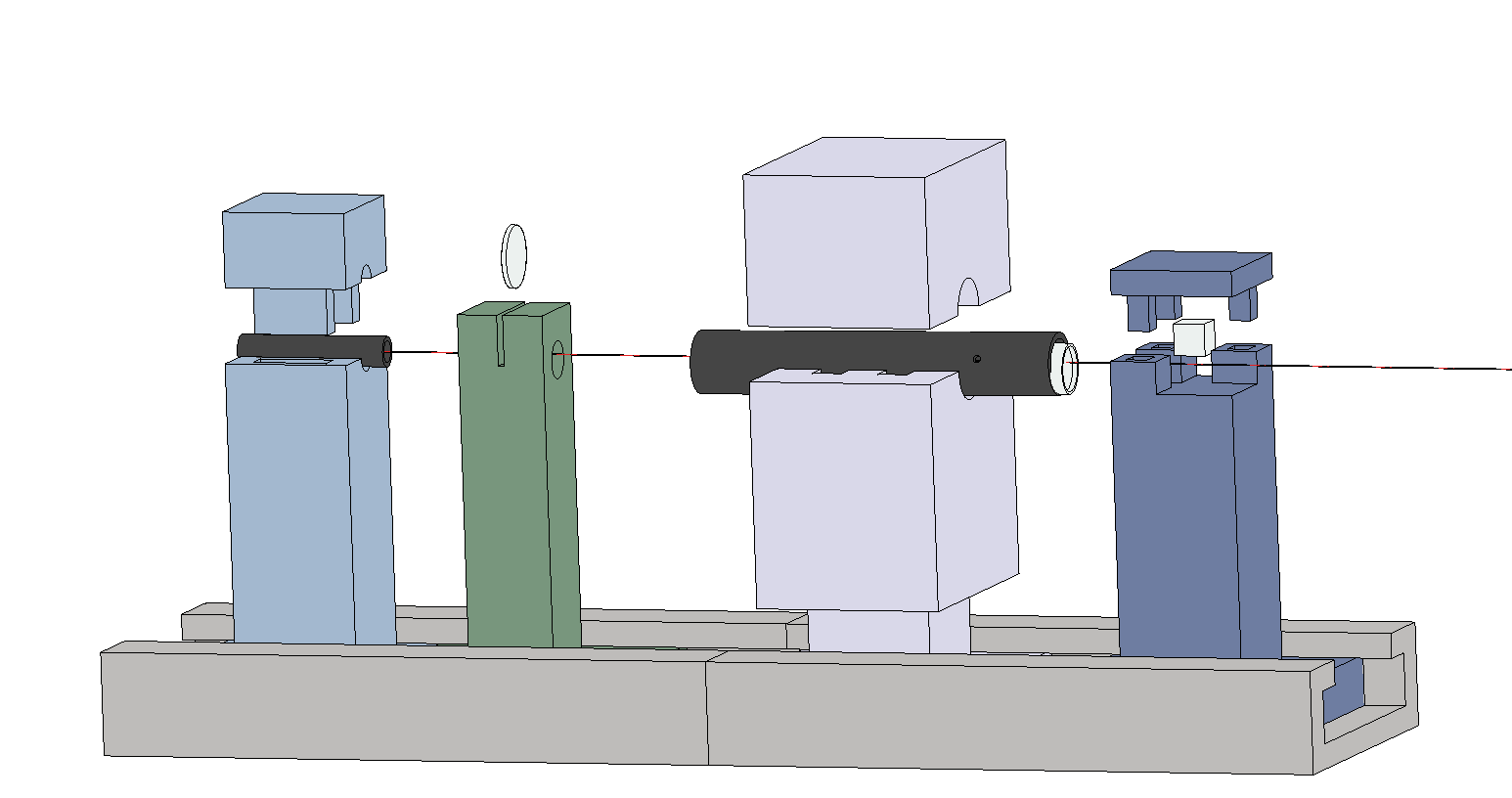
**User Guide to the device**

**Components needed**

* Power meter and a photodiode power sensor: Reads the laser power refracted from the beam cube
* Optics:
  + Polariser: linearly polarise the light in case the laser is elliptically polarised
  + Half Wave Plate: Changes the polarisation angle of the linearly polarised light
  + Beam Cube: Separates the vertical and horizontal components of the incident light
* Access to a 3D printer and laser cutter to print and cut the stability equipments and gears
* Motor Control:
* Raspberry Pi (you can also use an Arduino UNO)
* Stepper motor with 0.9° increments
* Motor driver (for example A4988)
* Breadboard, PCB or stripboard and wires

**Mechanical Stands:**

1. Open the CAD models of the stands
2. Dimension the stand parts such that they can fit your optics’ dimensions
3. 3D print the stand elements
4. Buy an aluminium tube of the same outer diameter than the tube holder (D). Make sure that the inner diameter of the aluminium tube can hold the half wave plate. The default design required a 20mm outer diameter.
5. Drill at the end of the tube (where the Half wave plate is positioned), three equidistant holes. Place some small screws to hold the half Wave Plate (the Half wave plate must be already mounted
6. Laser Cut the gears and the main stand using at least a 5mm plastic (acrylic, polypropylene). You can also glue two identical pieces of smaller diameter.
7. Mount the system according to picture 1



*Picture 1: Mounted mechanical stands*

*From left to right (A) Laser holder(B) Linear Polariser holder (C) Half Wave plate holder (tube) [and (D) its mount] (E) Polarising Beam Cube holder*

**Software:**

**Download the codes**

Clone the Github repository to download the code on your computer. The ‘Final’ branch is the latest developed. Some areas still need to be developed and have not been fully tested (PID loop) as the motor was most likely faulty during the tests.

Several libraries need to be added to use of the software:

* If you are using a Thorlabs PM100D power meter, download the ThorlabsPM100 package and follow the instructions: <https://pypi.org/project/ThorlabsPM100/>. This package requires the pyvisa package to control the instrument: <https://pyvisa.readthedocs.io/en/stable/>, and the National Instrument Visa Library, <https://www.ni.com/engb/support/downloads/drivers/download.ni-visa.html#346210>
* The remaining libraries used are usually provided when downloading Python

**Information on the code**

Connecting the Power Meter

* The Power Meter must be plugged in the computer via USB. The class ‘PowerMeter’ in GetPower.py is used in the main code to initialise and read values from the power meter. The example is given for Thorlab’s PM100 power meter. This class is abstracted, hence, you can easily change the initialisation to use a different device by just updating the class to meet your power meter’s requirements.
* The function \_\_init\_\_(self) initialises the device.
* usb\_port = rm.list\_resources() reads the components plugged to your computer and rm.open\_resource(usb) opens them.
* self.power\_meter = ThorlabsPM100(inst=inst) creates an instance of the power meter object with as power meter, the one opened from the USB.
* The function *readPower* reads the power from the power meter by calling the power meter’s library functions.

Running the program

* *App3.py* is the main function of the code. It calls the required libraries and packages used for the functioning of the device.
* The User Interface was designed using the library Tkinter. Threading is used to start and stop the device when the respective buttons are clicked.
* The timed data from the power meter are then saved in a CSV file.

Connecting the Microprocessor to the motor

We have not been able to test this properly as the motor or the motor driver were faulty during the experiments. The connection codes are in *ConnectionTestArduino.py* (Arduino) and *ConnectionR.py* (Raspberry Pi)

In *ConnectionTestArduino.py*, change the name of the port to the name of the port in your Arduino

In *ConnectionR.py c*hange in the class connection the UDP\_IP\_COMPUTER to the IP address of your computer, and UDP\_IP with the IP address of your Raspberry Pi (Which needs to be connected to internet)

Information to connect an Arduino to a motor and drive the motor from the Arduino can be found here: <https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/>