Introduction

In introductory optics usually starts with the geometric optics which assumes light travels in straight rays. In physical optics a wave model of light is used. One model used in physical optics is the Gaussian beam approximation for light propagation. The simplest Gaussian beam is a propagating beam with a circular amplitude intensity profile that is a Gaussian.

This project measures the horizontal intensity of a laser beam.

Background

A beam pattern is the *horizontal* dependence of the intensity of light. For example if you shine a laser beam on a wall and get close to it you will see something like *Figure 1*. Notice that the spot is brightest in the center and it is circularly symmetric. We need a function that describes this intensity pattern. A function we will find useful is a *Gaussian* profile for the electric field intensity is

$$E_s(r)=E_0\exp\left(-\frac{r^2}{w_0^2}\right)$$

where r is the radius from the origin, w_0 is the diameter at 1/e of the maximum amplitude, and E_0 is the electric field amplitude. Note that where $r=w_0$, $E_s(w_0)=E_0/e\approx 0.37\,E_0$

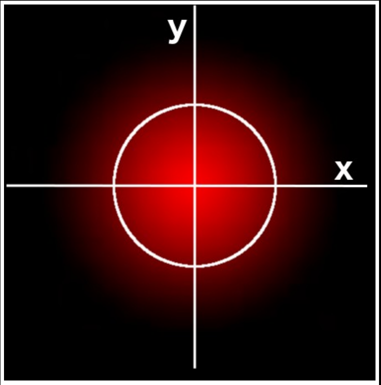


Figure 1: Close up of laser spot.The white circle shows the 1/e amplitude contour.

A propagating wave with a Gaussian profile is a solution to Maxwell's Equations, so this profile is fully consistent with the wave nature of light.

From ray optics you know that if you put a positive lens in collimated beam (and a laser beam is collimated) it focuses down to a point one focal length from the lens. How is this described using a Gaussian beam?

You can solve Maxwell's equations for a converging Gaussian beam. The result is that the beam diameter gets smaller and smaller, but instead of converging to a point, it reaches a minimum diameter at the focal point of the lens (called the *beam waist*), then starts diverging. So the basic picture from ray optics is still true, but the wave nature of light modifies the result. *Figure 2* show a section of a Gaussian beam as it goes through a focal point. The red curve shows the 1/e contour versus distance along the propagation direction.

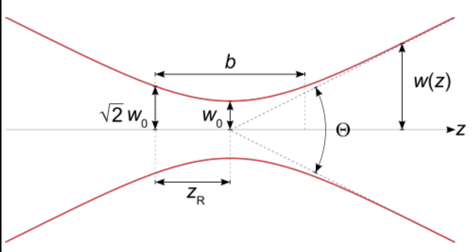


Figure 2: Gaussian beam width w(z) as a function of the distance z along the beam.The 1/e amplitude contour is plotted.

The dashed lines show the non-wave picture: the light would converge to a point in a cone, then diverge with the angle Θ being the convergence and divergence angle. When the wave nature of light is taken into account, you get the "smoothing" over a distance characterized by b and a minimum radius of w_0. The beam radius as a function of z is

$$w(z)=w_0\sqrt{1+\left(\frac{\lambda z}{\pi w_0^2}\right)^2}$$

where λ is the wavelength of light and z is defined as zero at the beam waist. It is remarkable that the whole profile is determined by only two parameters: the wavelength and the waist diameter. The *Rayleigh range* is defined by

$$z_R = \frac{\pi w_0^2}{\lambda}$$

At a distance $\pm z_R$ from the beam waist, the diameter of the beam is $\sqrt{2} w_0$ and $b = 2 z_R$. The last parameter of interest is the asymptotic angle. It can be derived as

$$\Theta = 2 \frac{dw}{dz} \approx \frac{2\lambda}{\pi w_0} \quad .$$

Measuring The Beam Profile

A traditional way of measuring a beam profile is to mount a razor beam on a translation stage and measure the intensity of the laser beam versus position, *Figure 3*. As the razor blade cuts off the beam the intensity has a shape that depends on the beam profile, *Figure 4*.

This is an experiment crying out for a modern microcontoller interface. In this project you will build a microcontoller controlled translation stage with a digital light sensor.

The translation stage is 3D printed with about $2 of filament.

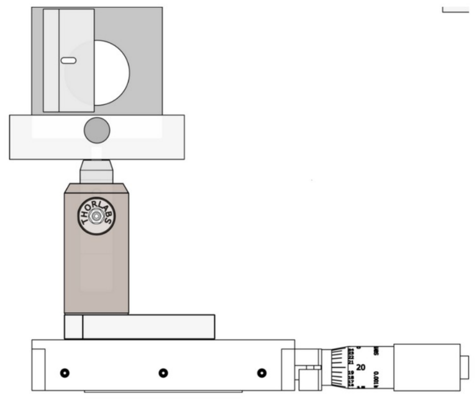
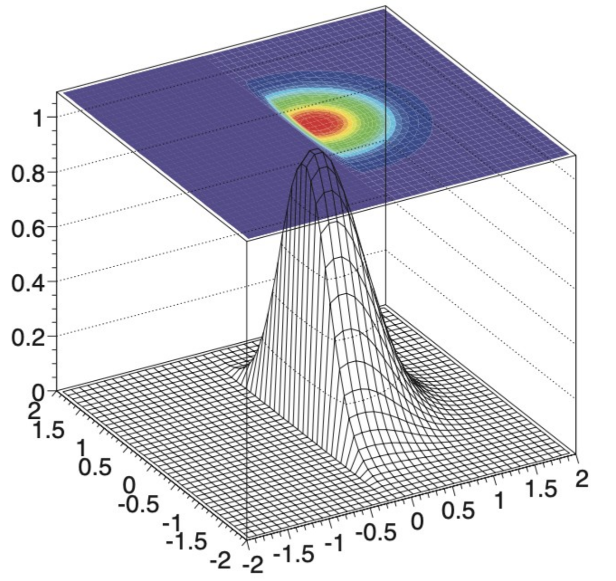


Figure 3: Traditional laser beam profile measurement.



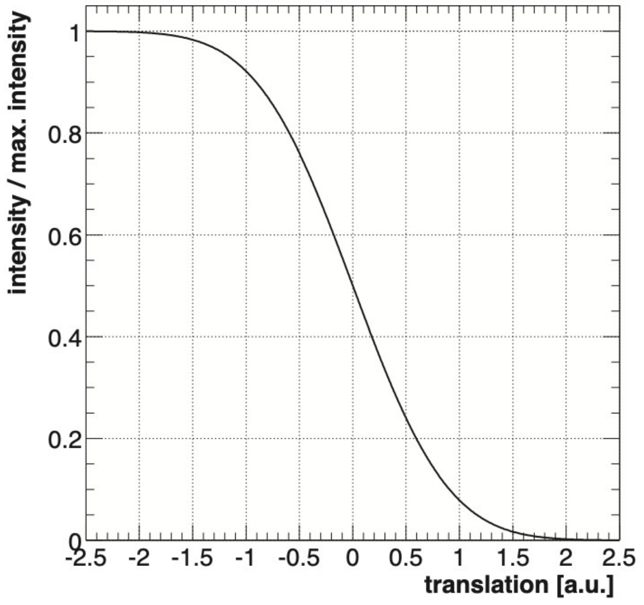


Figure 4: The intensity relative to the maximum intensity as function of the translation of the blade into the laser beam

BUILDING THE PROJECT

Bill of Materials

- MotorizedPrecisionTranslationStage_22a.scad, 3D print
- Translation stage spacer
- Gel Super Glue
- Hex head screw, M3 x 30 mm
- Rubber band
- 28BYJ-48 stepper motor
- 2: M3-0.5 x 6mm thread-forming screws, Fillister-Phillips.
- VEML7700 light sensor breakout board.
- Four XXX XXX thread-forming screws.
- 3D printed case
- Half-breadboard
- XXX Motor driver breakout board
- Six color of hook up wire

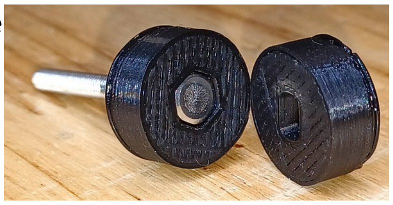
Tools

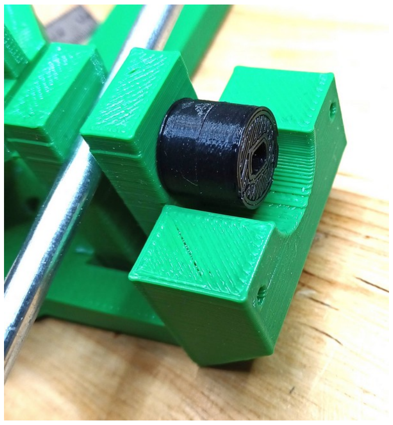
- Toothpick
- Wire cutters
- Spacer rod
- Small regular screwdriver
- Small Phillips screwdriver, PH1.
-

The Translation Stage

The translation stage is basically one piece of 3D printing. The moving stage is a parallelogram with thin flexures at the tops and bottoms of the vertical arms. There is one arm for mounting a stepper motor, and a second, longer arm for mounting a light sensor. To assemble the translation stage:

- Cut away the eight thin vertical columns. These are to support the filament during the printing of the upper stage. You may have to trim other filament strands if they interfere with the movement of the stage.
- Glue the octagonal spacer to the bottom of the stage. This is to provide clearance for the flexing of the vertical arms. Gel superglue works well.
- Insert the hex screw into the screw capture disk with a hexagonal socket, then glue the other capture disk to the assembly, *Figure 5a*.
- (Optional) Drill out the screw clearance hole with a 3mm or 7/16” drill.
- Insert drive screw. Work stage back and forth to loosen the screw in the holes.
- Seat hex nut in hex socket in vertical arm. It can be a tight fit. You can insert a spacer rod and tighten the screw by turning the screw capture disk to seat hex nut firmly in socket, *Figure 5b*.
- The clearance holes for the drive screw are initially tight. With the crew in place and threaded through the nut, work the top of the stage back and forth to loosen the screw in the holes.
- Loop rubber band over interior post, then string both strands around the moving vertical arm, and loop them back over interior post from other side. This keeps tension on the nut.
- Test movement by turning the nut capture disk both directions. The top of the stage should move both directions.
- Mount the 28BYJ-48 stepper motor to the stage with the D shaft in the capture disk. Secure with two 3 mm thread-forming screws.
- Mount the VEML7700 light sensing board to the long arm using XXX.
- I taped a piece of a cut up ping pong ball over the sensor to act as a diffuser. There was about 2 mm between the sensor and the diffuser.


a)Nut capture disk.


b)Tightening screw with spacer rod.

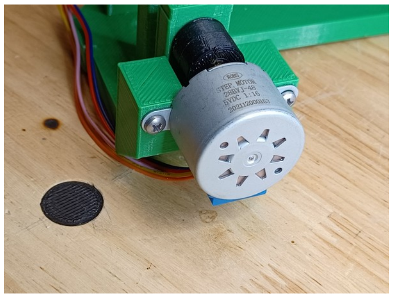

c)Mounted motor.

Figure 5: Assembly pictures.

Making the Device

This project has two breakout boards, the motor driver and the light sensor. Both are made by AdaFruit, so they have support in CircuitPython. This project is an example of two sensors using the same I2C bus.

Assemble the Case and Breadboard

- Peel the strips off of the half-breadboard and stick it inside the case. Make sure the tabs on the breadboard are to the left and bottom. Center the board right to left and against the bottom of the case. There has to be room on the right and left for the snap tabs to fit inside the bottom of the case.

- Mount the S2Mini at the top of the breadboard. See *Figure 6*.
- Mount the AdaFruit Motor FeatherWing at the bottom of the breadboard with the two power terminal at the bottom.

Wire the Device

Note: The reference for wiring the different pieces should be the respective data sheets.
Refer to the figure on the next page when wiring the device.
S2Mini Connections:

- The motor is a 5 V motor, so it should be powered directly off of the USB voltage. The motor driver board has separate logic and power circuit, as most driver boards do**.** Wire the S2Mini VBUS pin to the red (+) power bus on the left side of the breadboard.
- Wire the S2Mini GND pin to the negative (-) power bus.
- Wire the S2Mini 3.3V pin to the left (+) power bus.

The next two connections are shared by both the motor driver and the light sensor boards.

- Wire S2Mini pin SDA 33 pin to row 12 on the breadboard.
- Wire S2Mini pin SCL 35 pin to row 11 on the breadboard.

Motor Driver Connections

- Wire the (+) screw connector to the left (+) power bus.
- Wire the (-) screw connector to the left (-) power bus (ground).
- Wire the left (-) power bus to the GND pin on the motor driver board. This is row 25 on the right side of the board.
- Wire the right (+) power bus the 3.3V pin on the motor driver, row 29 on the right.
- Wire the motor driver SDA pin (row 15) to the previous SDA row 12.
- Wire the motor driver SCL pin (row 16) to the previous SCL row 11.

Light Sensor Connections

- Use a Male to Female jumper wires for these connections.
- Wire the light sensor VIN to (+) 5V left power bus.
- Wire the light sensor GND to (-) GND left power bus.
- Wire the light sensor SCL to the SCL row 11.
- Wire the light sensor SDA to the SDA row 12.

The Stepper Motor

The stepper motor is connected to the four screw terminals on the end of the motor driver board. I number the connections 1, 2, 3, and 4 from left to right.

- Screw terminal 1 connects to the pink wire of the stepper.
- Screw terminal 2 connects to the orange wire of the stepper.
- Screw terminal 3 connects to the blue wire of the stepper.
- Screw terminal 4 connects to the yellow wire of the stepper.

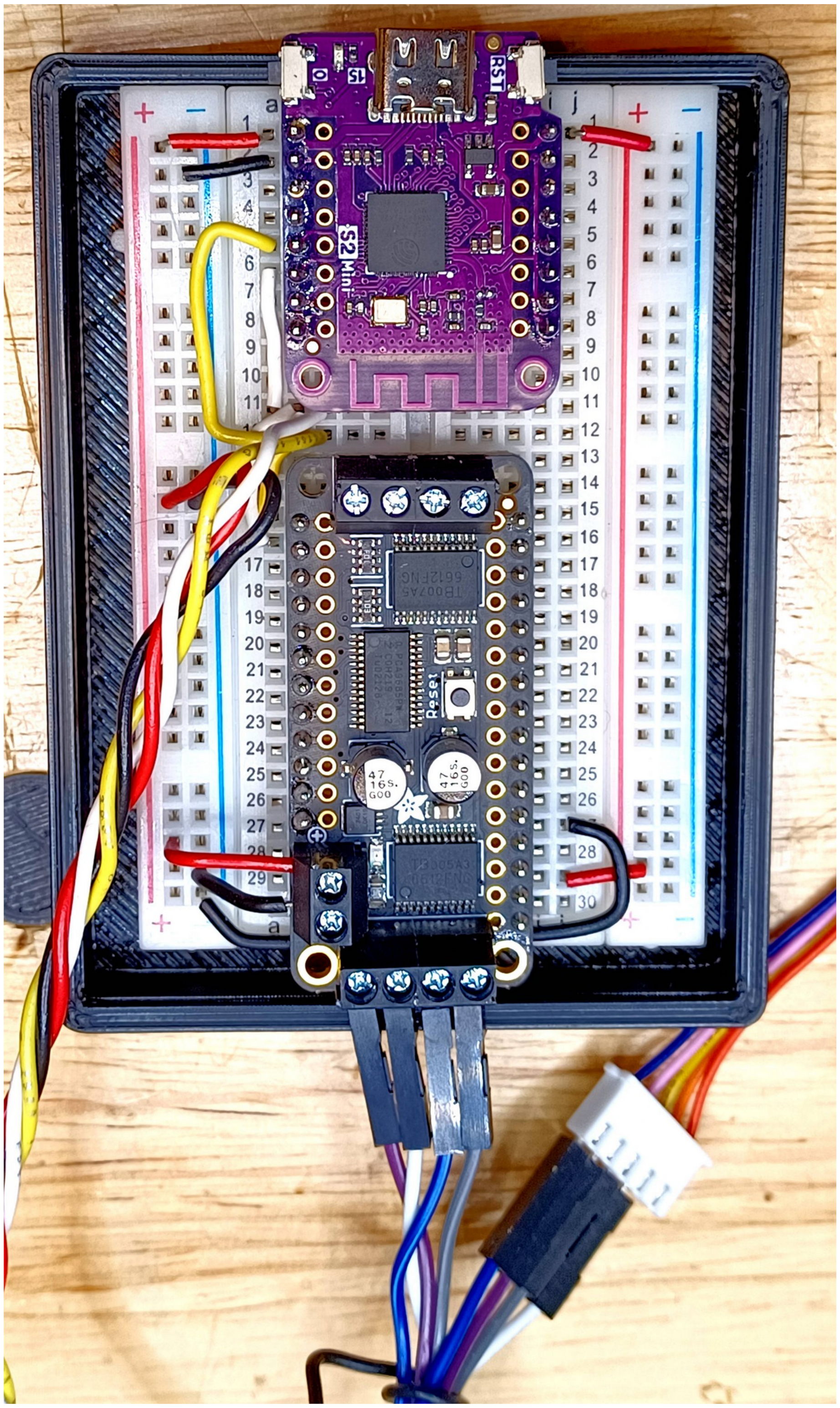


Figure 6: Picture of wired device.

SOFTWARE

Testing the Light Sensor

The light sensor outputs the light intensity in three forms, **`light`** which is the raw counts from the sensor, **`white`** which is spectrally compensated to mimic the human eye response, and **`lux`**, a standard unit. The sensor has a 16-bit digitizer so the counts output ranges from 0 to 65535. The sensor also has different gain and integration times that the user can set. The sensor also can output the estimated resolution of the lux readings. I wrote a test program, **`testLuxSensor_22b.py`**, to read the light sensor output for different values of these parameters. The code also averages over 10 readings to give an estimate of how stable each setting is. This code is a good example of how to use the VEML7700 light sensor.

I tested the sensor in two cases: room light, file **`2022-06-14-testLuxSensor-dim light.csv`** and with a Class 2 (< 1 mW) laser shining directly on the diffuser., file **`2022-06-14-testLuxSensor-laser-diffuser.csv`**. For the room light, the lux readings were very consistent across the gains and integration times, even when the counts were at the low end of the range. The highest counts, 7529,were at a gain of 2 and an integration time of 800 ms.
In contrast, for the laser many of the readings saturated the counter. Based on these results I set the default gain to 1.0 and integration time to 25 ms which gave about 11000 counts. These values can be set in the Laser Profiler software.

Operating Software

To run the device I programmed a set of SCPI-like commands so the operator can set parameters for the data run and take data. When you run the code it prints the menu of functions. At the end of a data run, I cut and paste the data into a text editor, edit out the non-data lines and save it as a CSV file. Here is the help menu:

```
>> help
DATA - Takes 0 data points. Moves 0 steps every point.
DSTP [#?] - Set or return number of steps per data point.
GAIN [#?] - Set or return gain. Set must be from (0.125, 0.25, 1.0, 2.0).
HELP - Prints this list.
HOME - Go to Home position.
IDN - Identify software and hardware.
INTT [#?] - Set or return integration time. Set must be from (25, 50, 100, 200,
400, 800).
LUX - Prints light sensor reading in lux.
NDAT [#?] - Set or return number of data points.
POS - Prints current position in step and microns.
QUIT - Exit to the REPL.
RST - Resets default parameters.
SETH - Set Home to current position.
STEP # - Move stepper # steps. Negative is backward.
STUM # - Move # microns rounded to nearest step. Negative is backward.
```

Note: the commands are not sensitive to case. I usually use lower case.
The commands that are high-level functions are:

- HELP - Prints all commands with a brief description of each one.
- IDN - Identify software and hardware. This is a standard SCPI command.
- QUIT – Stops the code and gets you to the CircuitPython prompt **>>>**.
- RST - Resets default parameters. These are the home position, number of data points, steps per data point, the light sensor gain, and the light sensor integration time.

The next logical group are the positioning commands:

- STEP # - Moves the stage the number of steps in the command. Mine is wired up so positive is to the right as you look from the stage toward the light sensor. Negative is opposite. Also note that at the end of movement in the negative direction, the stage over shoots and takes out backlash by taking positive steps.
- STUM # - Move an approximate number of microns. These are rounded to the nearest step.
- SETH – Set the current position to the home position.
- HOME – Go to the home position.
- POS – prints the current position out in number of steps and microns.

The next group of commands operate the light sensor.

- LUX – prints the current light reading in lux.

Note: the notation [#?] is that one of the two options, a number or a question mark should be given to the command.

- GAIN [#?] - Sets or reports the light sensor gain. A ? means the current gain will be printed out. A number attempts to set the sensor gain. The gain must be once of the values (0.125, 0.25, 1.0, 2.0).
- INTT [#?] - Sets or reports the light sensor integration time. The integration time is in milliseconds and must be one of the values (25, 50, 100, 200, 400, 800).

Finally the last group of commands control taking data.

- NDAT [#?] - Sets or reports the number of data points to be taken in a run. The stage moves between data points.
- DSTP [#?] - Sets or reports how many steps should be taken between data points. It only steps in the positive direction to avoid backlash issues.
- DATA – Takes the data. The data is printed out in CSV format. Comments start with a #, and the fields are separated by commas. Here is a small run:

```
>> ndat 11
>> dstp 10
>> data
0, 0.0, 8805, 31.698
10, 10.928, 8805, 31.698
20, 21.856, 8758, 31.5288
30, 32.784, 8758, 31.5288
40, 43.712, 8758, 31.5288
50, 54.64, 8758, 31.5288
60, 65.5679, 8758, 31.5288
70, 76.4959, 8758, 31.5288
80, 87.4239, 8818, 31.7448
90, 98.3519, 8818, 31.7448
100, 109.28, 8818, 31.7448
>>
```

- The fields are:
  - The position in steps.
  - The position in microns.
  - The raw light counts.
  - The lux value.
- I would cut and paste the data into an editor and save it with a `.csv` file suffix. CSV files can easily be read by python, R, Excel, and many other programs.

Sample Data and Analysis

Example of data and fit to a Gaussian beam.

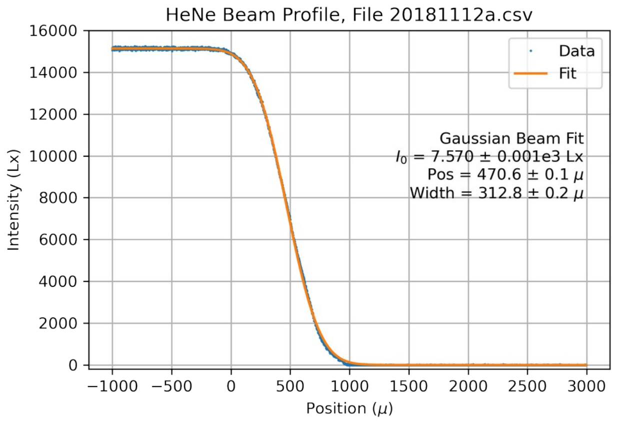


Figure 7: Data and best fit.

The fit residuals.

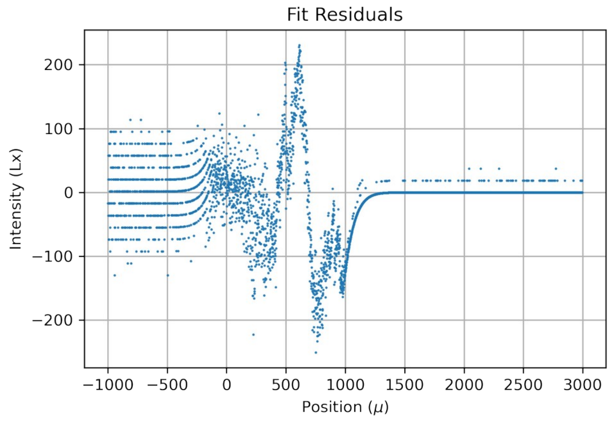


Figure 8: The fit residuals.