Lab 2: Wave-Particle Duality

Kevin Nelson, Jianming Qian, Alexander Takla
Michigan Math and Science Scholars
26 July 2023

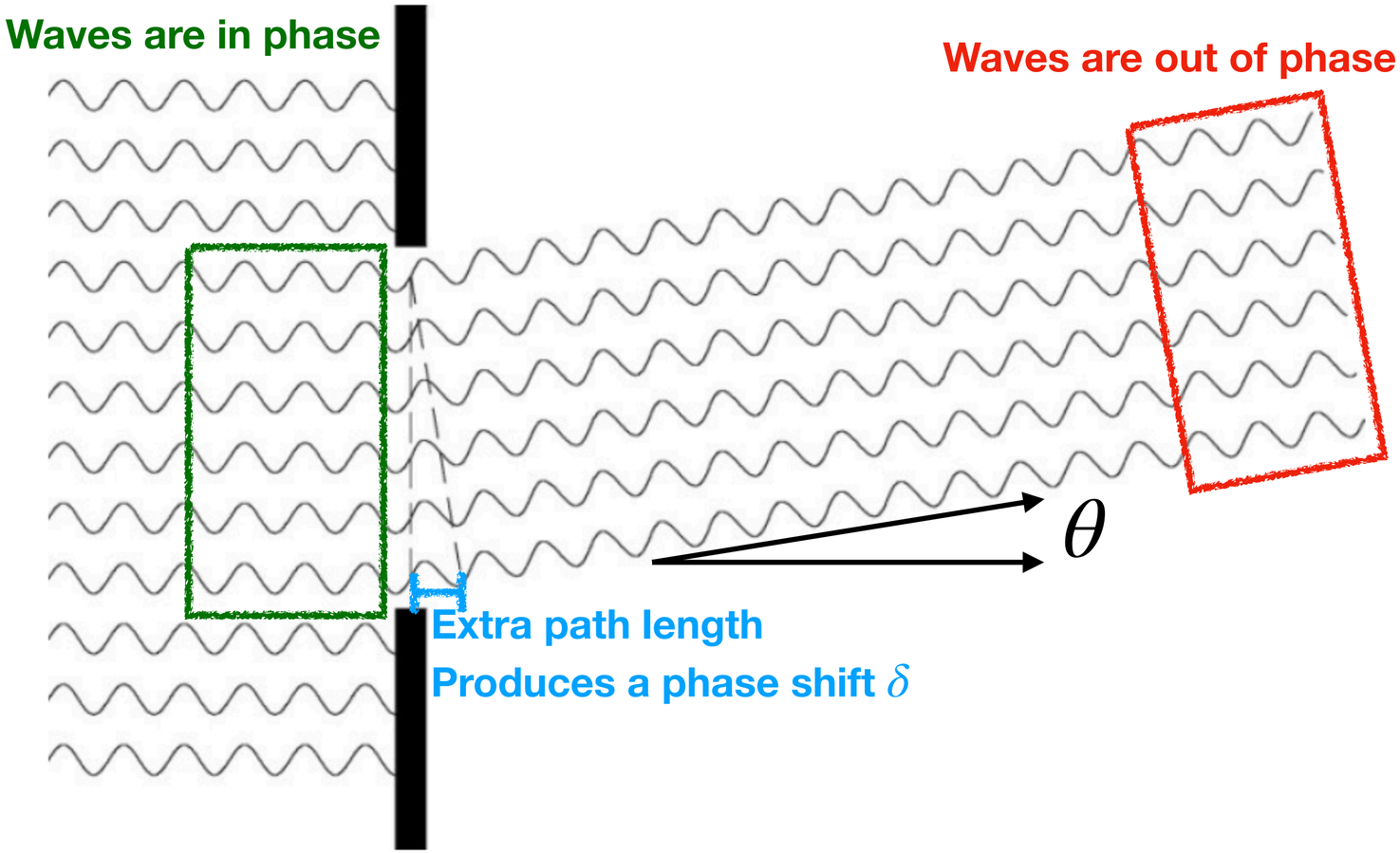
Light Waves

- **You may be familiar with two concepts:**
  - 1. Light is “a wave”
  - 2. Light is made up of individual “photons” or “light particles”

- **So which is it? Is light a wave or a particle? Today we will see that it is *both*.**

Diffraction

- **Diffraction occurs when light goes through a thin opening**
- **Constructive and destructive interference occurs as a function of the angle θ**
  - In phase = constructive interference
  - Out of phase = destructive interference
- **How do the waves go from in phase to out of phase?**
  - Extra path length increases with θ



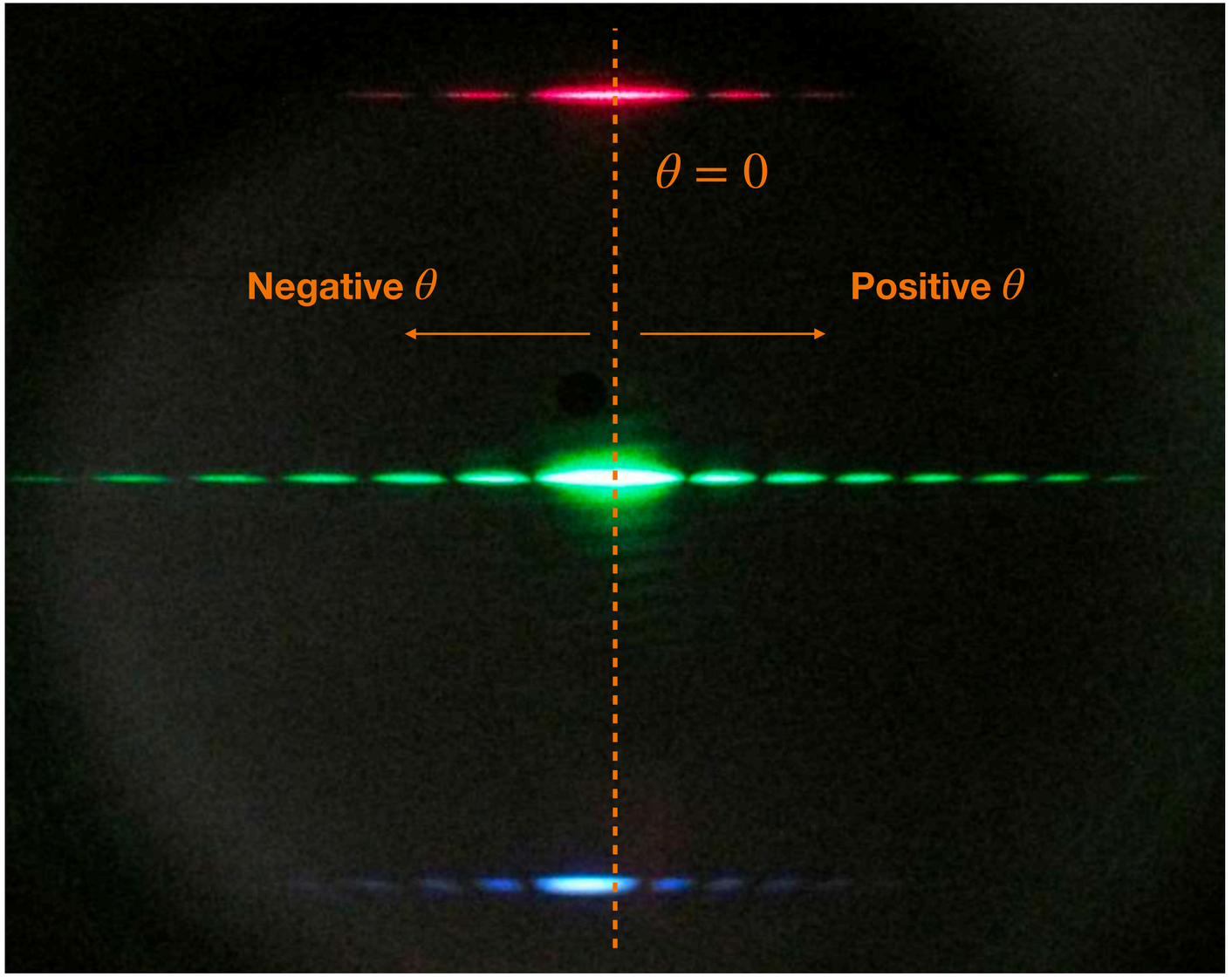
Diffraction patterns

- **An interference pattern is produced when we shine light through a small slit**
- **The locations of the dark bands are:**

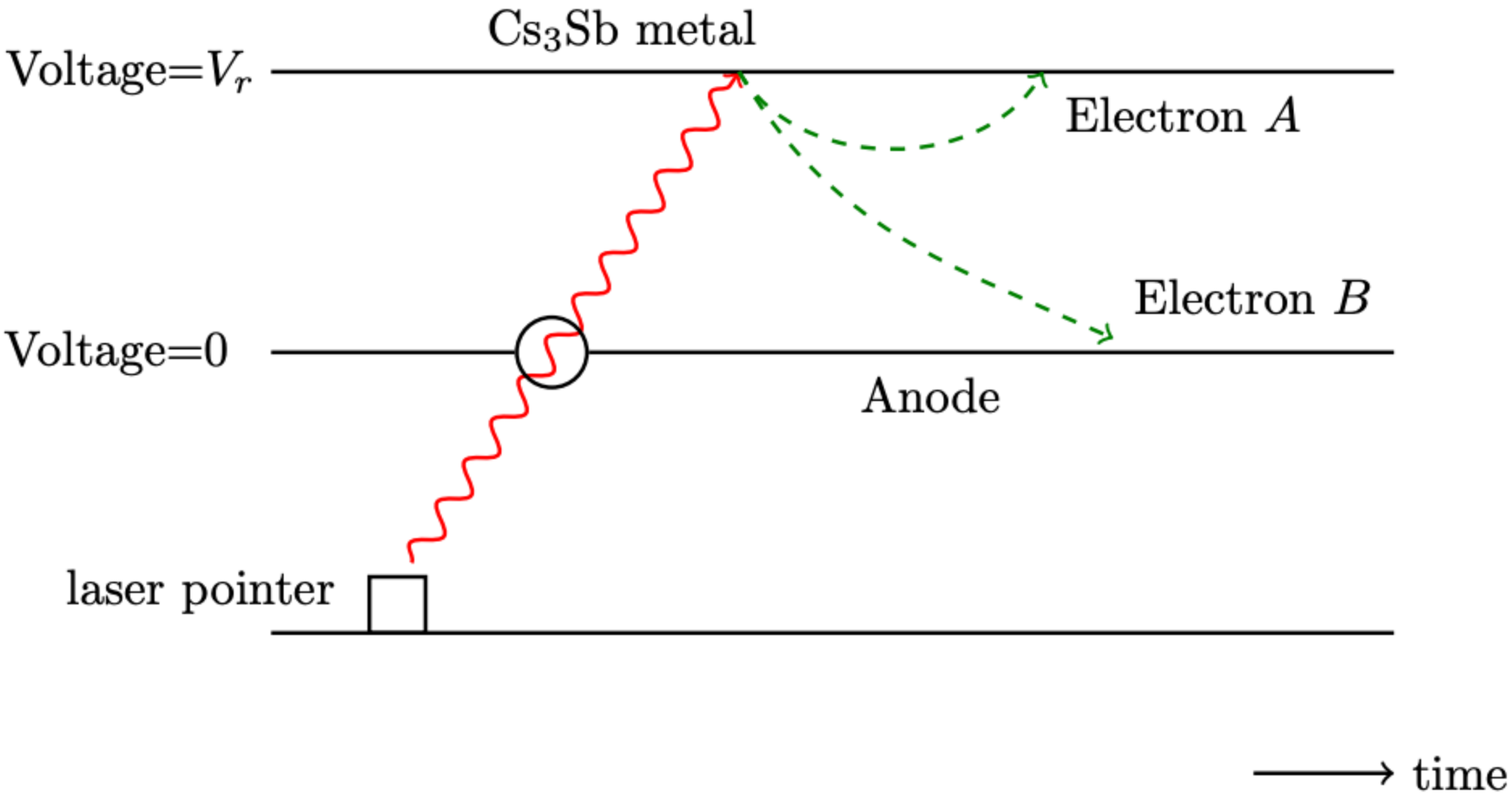
$$\sin\theta = n\frac{\lambda}{a}$$

  - The sine means we expect a periodic interference pattern

- **You will verify the mathematical relationships between wave parameters and interference patterns in the first part of the lab**
- **This interference demonstrates that light is a wave!**

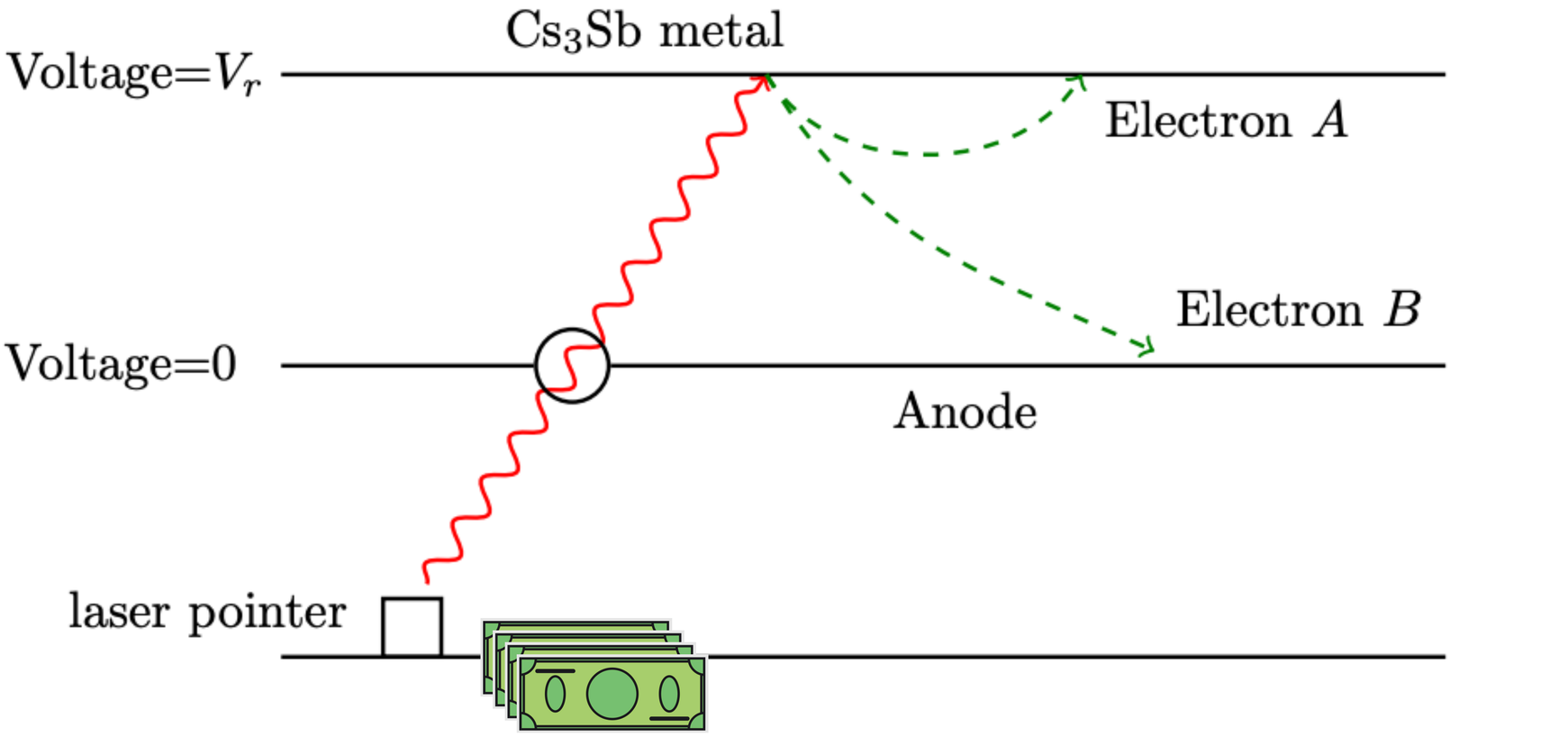


The photoelectric effect



The photoelectric effect

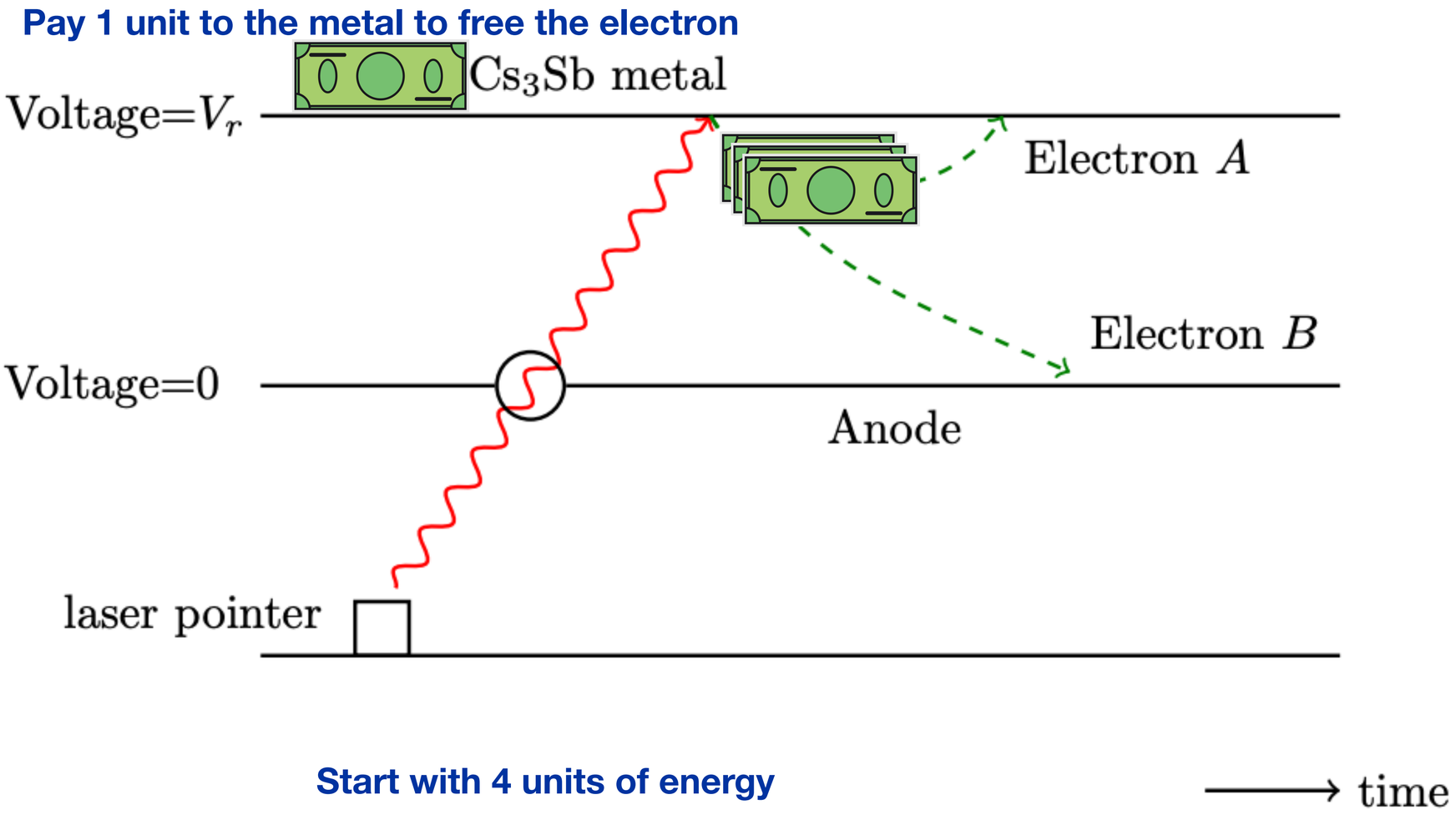
- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)



Start with 4 units of energy

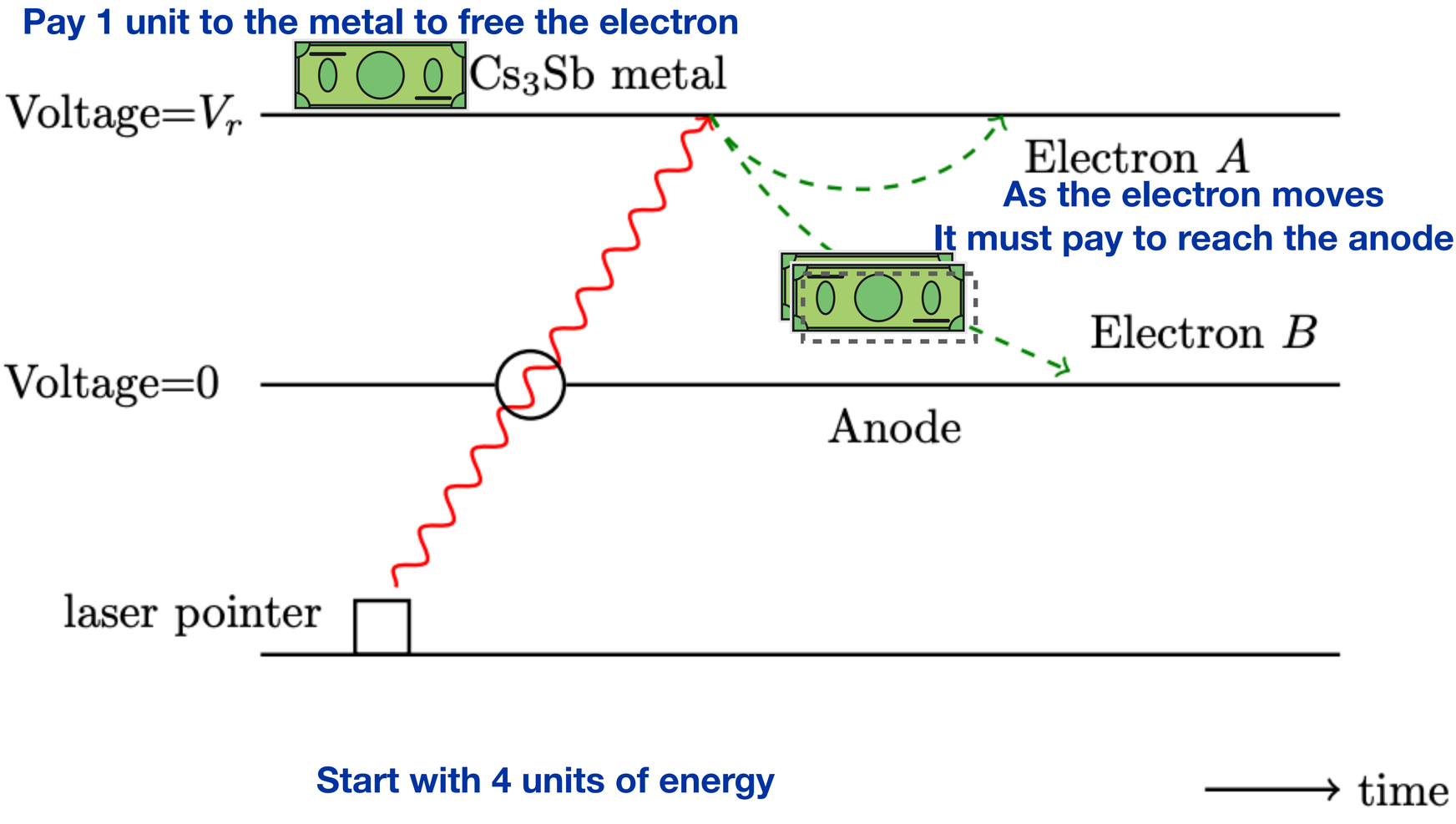
The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.



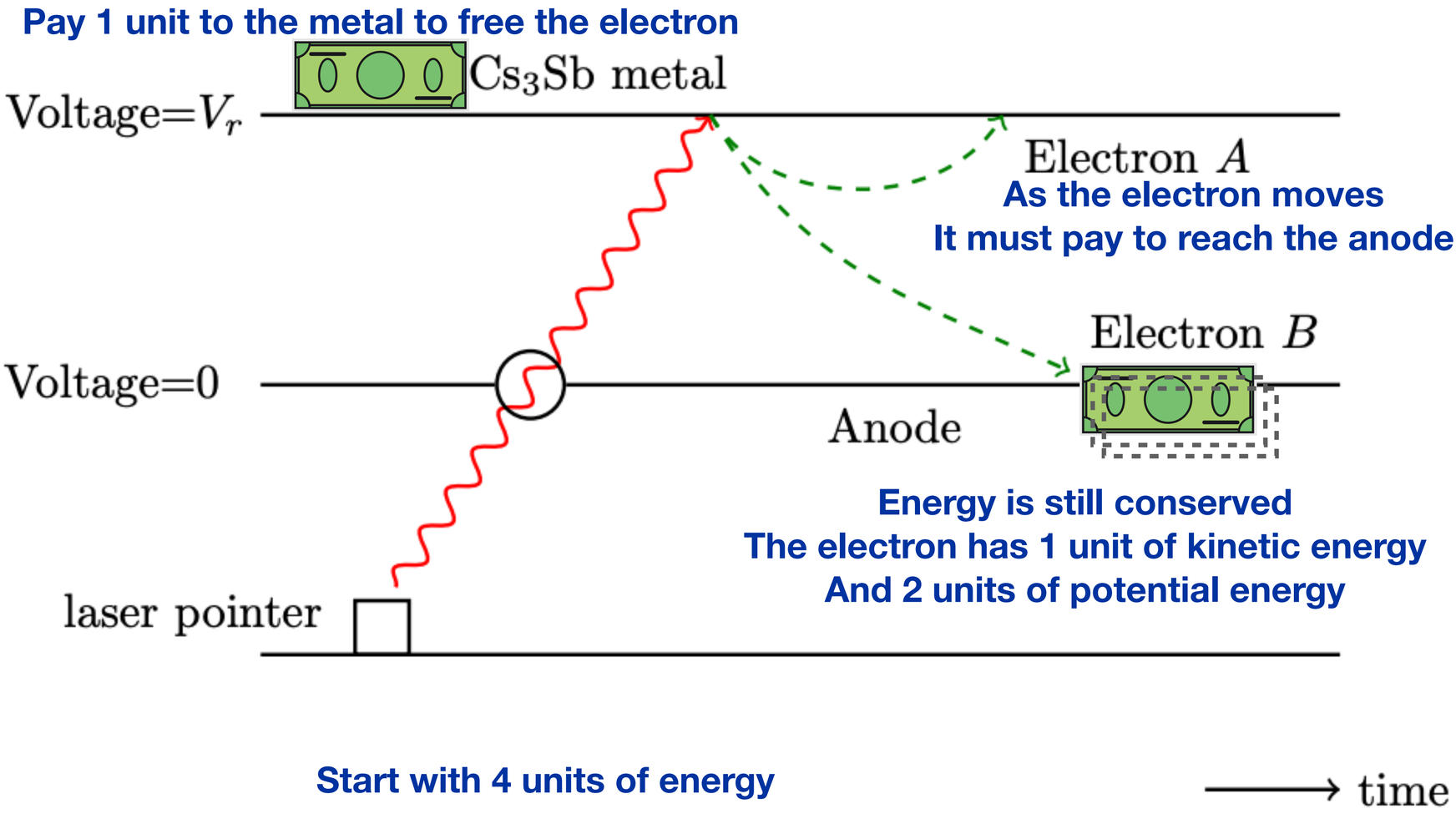
The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



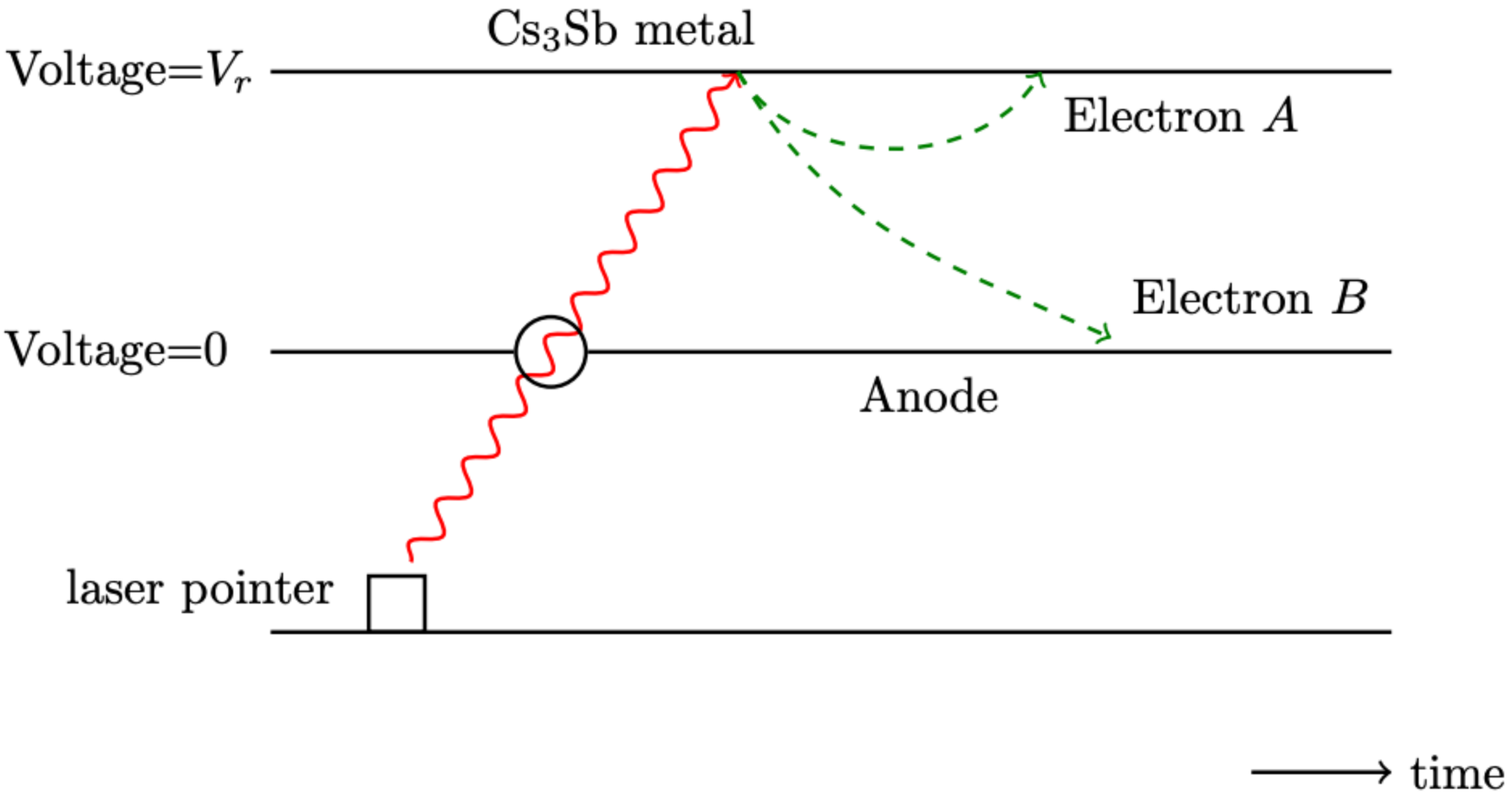
The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



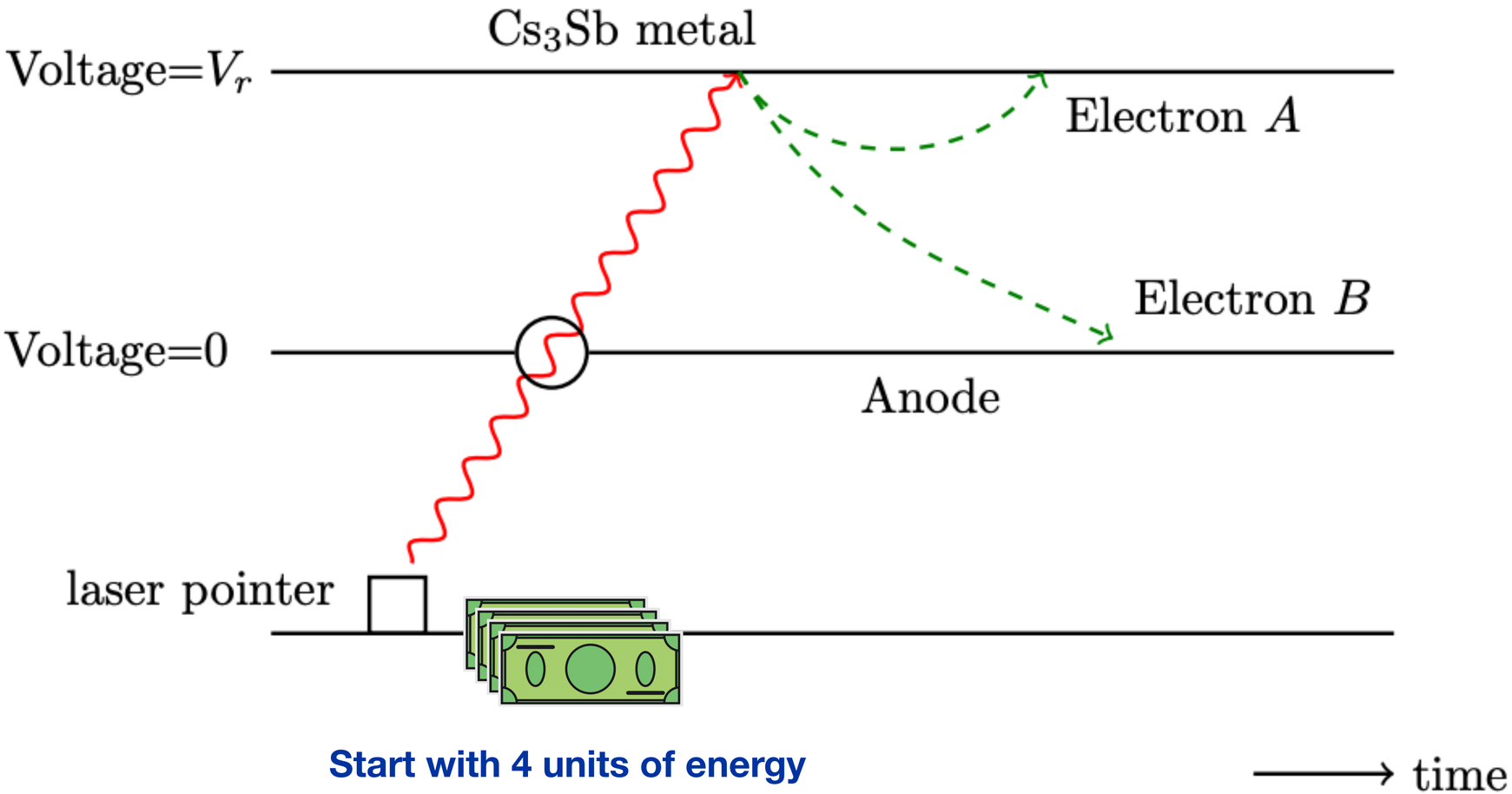
The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



The photoelectric effect

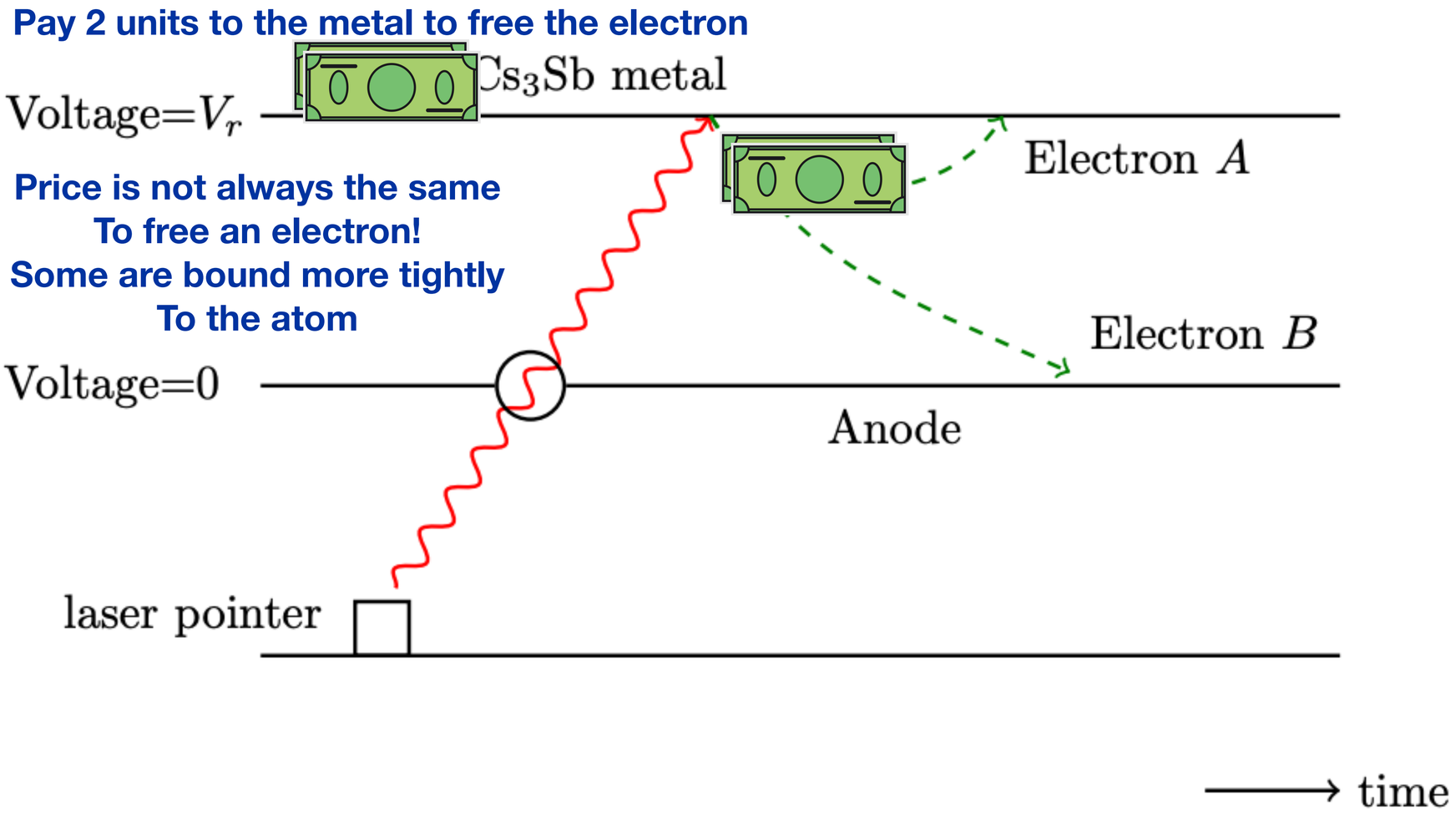
- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



Kevin Nelson

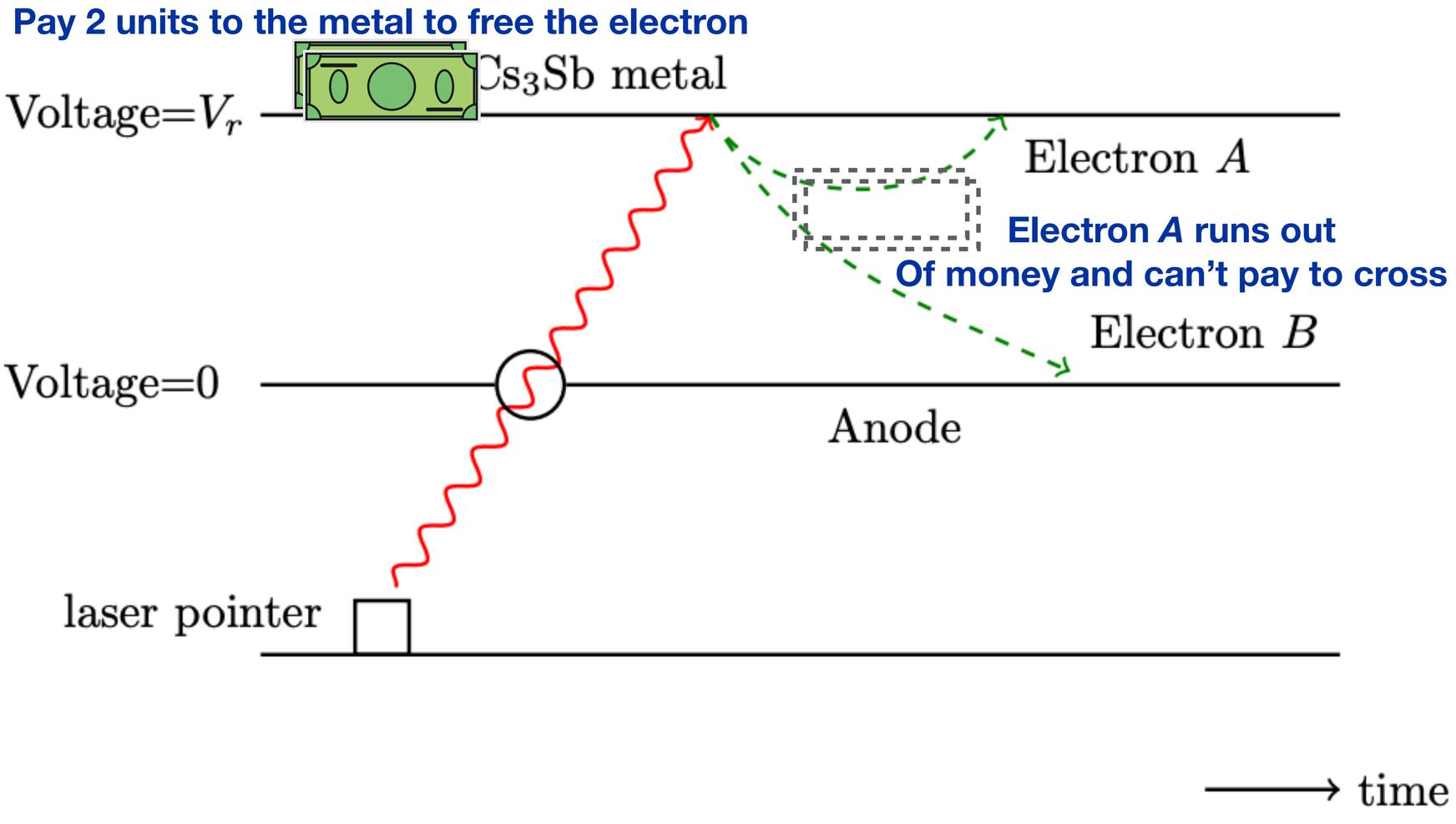
The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



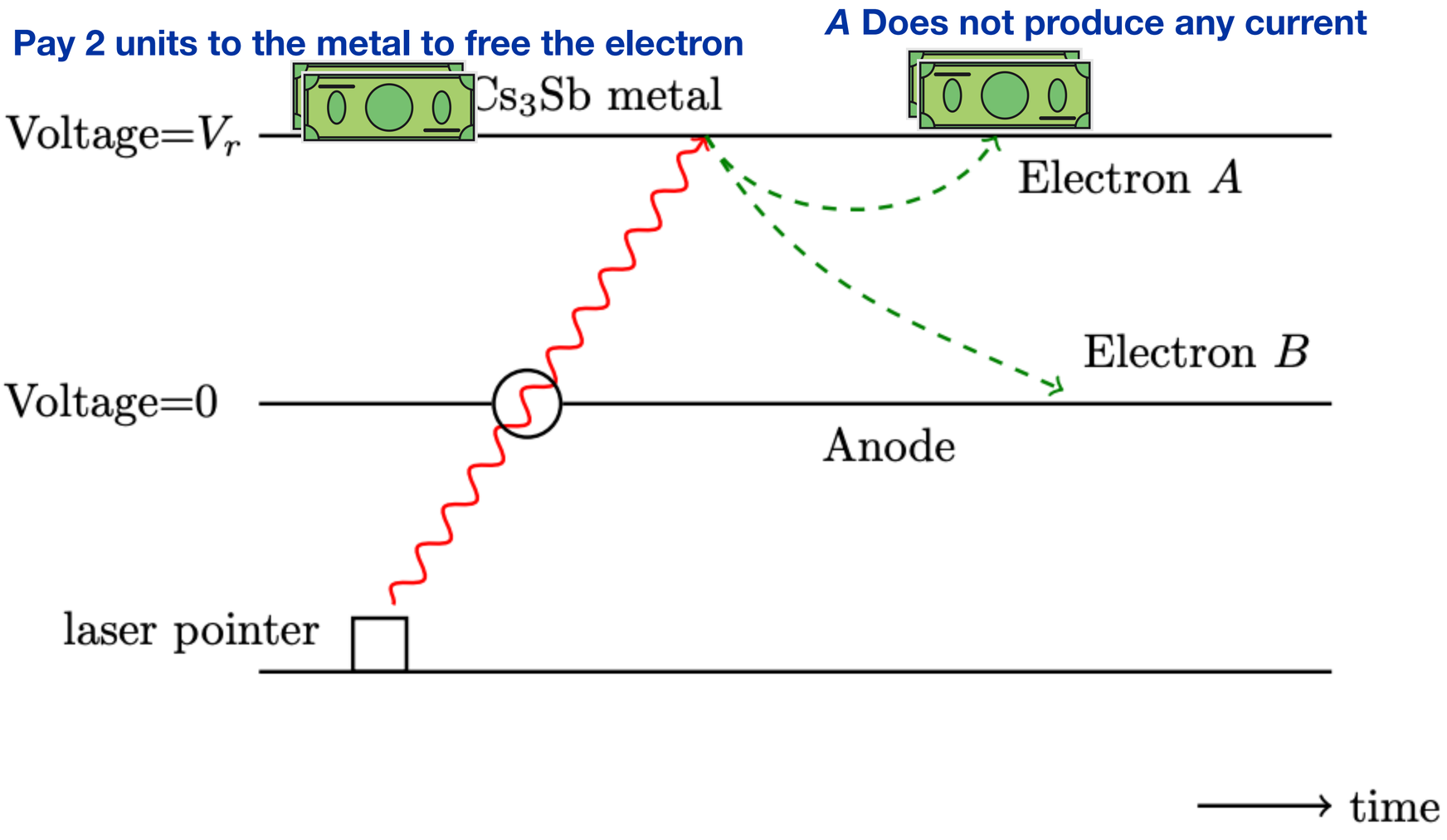
The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



The photoelectric effect

- **Photons from the laser pointer shine onto a metal surface**
  - Photons start with a certain amount of energy (represented as money)
- **The photon can have enough energy to kick an electron out of the metal**
  - An energy price must be paid to free the electron. Electron is bound with some binding energy.
- **The electron is repelled from the Anode**
  - Electron A does not have enough energy to overcome the voltage barrier and so it does not produce any current
  - Electron B has enough energy to pay the voltage price and still reach the anode



The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

- **The equation governing this effect is very simple**

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

- **The equation governing this effect is very simple**
  - E_{max} is the maximum energy an electron can have after being freed by the photon

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

- **The equation governing this effect is very simple**
  - E_{max} is the maximum energy an electron can have after being freed by the photon
  - h is Planck's constant, representing the quantization of energy. We will measure this in the lab today

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

- **The equation governing this effect is very simple**
  - E_{max} is the maximum energy an electron can have after being freed by the photon
  - h is Planck's constant, representing the quantization of energy. We will measure this in the lab today
  - f is the frequency of the light

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

- **The equation governing this effect is very simple**
  - E_{max} is the maximum energy an electron can have after being freed by the photon
  - h is Planck's constant, representing the quantization of energy. We will measure this in the lab today
  - f is the frequency of the light
  - ϕ is the minimum binding energy. Remember, not all electrons are bound with the same energy. Some are "closer" to the nucleus and need more energy to free. This number has units of volts.

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

- **The equation governing this effect is very simple**
  - E_{max} is the maximum energy an electron can have after being freed by the photon
  - h is Planck's constant, representing the quantization of energy. We will measure this in the lab today
  - f is the frequency of the light
  - ϕ is the minimum binding energy. Remember, not all electrons are bound with the same energy. Some are "closer" to the nucleus and need more energy to free. This number has units of volts.
  - e is the charge of the electron $= 1.6 \times 10^{-19} Js$

The photoelectric effect

$$E_{\mathbf{max}} = hf - e\phi$$

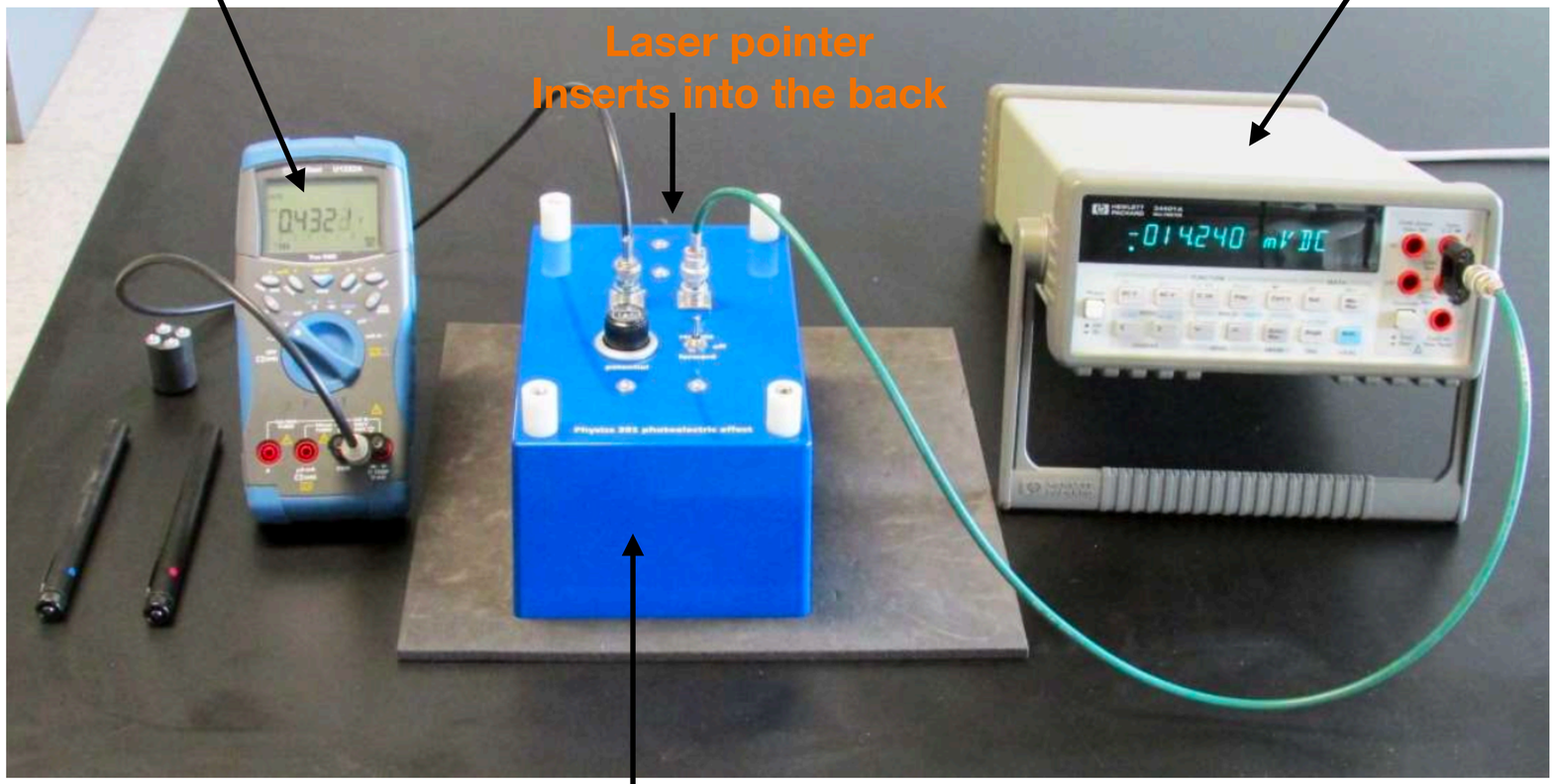
- **The equation governing this effect is very simple**
  - E_{max} is the maximum energy an electron can have after being freed by the photon
  - h is Planck's constant, representing the quantization of energy. We will measure this in the lab today
  - f is the frequency of the light
  - ϕ is the minimum binding energy. Remember, not all electrons are bound with the same energy. Some are "closer" to the nucleus and need more energy to free. This number has units of volts.
  - e is the charge of the electron $= 1.6 \times 10^{-19} Js$

- **Energy of freed electrons is not a function of light intensity, only frequency! This proves that light is made of *particles*!**

Experimental setup

Current monitor
Measure the photocurrent here

Voltage source
You set the value of the voltage here



Photoelectric effect
Occurs in this box

IMPORTANT: SET TO “REVERSE” MODE

Fitting data with equations

- **Today we will use our first *Juypter notebook* to analyze data with a computer**

- **This is a simple python script to fit some data with a function and plot the results**

In [33]:

```
'''
This code is used to set up the fitting library.  Run this block without modification.
'''

%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as opt

'''
Arguments:
  - func:   the callable function used to fit the data.
            First argument must be the indpendent variable, rest of arguments are free parameters
  - xdata:  list of x axis coordinates of the data
  - ydata:  list of y axis coordinates of the data
  - xlabel: the title to put on the x axis
  - ylabel: the title to put on the y axis

Returns:
  - optimizedParameters: a list of the best fit values of the function parameters
'''
def fit(func, xdata, ydata, xlabel='', ylabel=''):
    plt.plot(xdata, ydata, ".", label="Data")
    optimizedParameters, pcov = opt.curve_fit(func, np.asarray(xdata), np.asarray(ydata))

    xinterp   = np.linspace(min(xdata), max(xdata), 100)
    fitinterp = [func(x, *optimizedParameters) for x in xinterp]
    plt.plot(xinterp, fitinterp, label="fit")
    plt.legend()
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print('The best fit parameters are: ', [p for p in optimizedParameters])
    return optimizedParameters
```

Fitting data with equations

- **Today we will use our first *Juypter notebook* to analyze data with a computer**

- **This is a simple python script to fit some data with a function and plot the results**

In [33]:

```
'''
This code is used to set up the fitting library.  Run this block without modification.
'''

%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as opt

'''
Arguments:
  - func:   the callable function used to fit the data.
            First argument must be the indpendent variable, rest of arguments are free parameters
  - xdata:  list of x axis coordinates of the data
  - ydata:  list of y axis coordinates of the data
  - xlabel: the title to put on the x axis
  - ylabel: the title to put on the y axis

Returns:
  - optimizedParameters: a list of the best fit values of the function parameters
'''
def fit(func, xdata, ydata, xlabel='', ylabel=''):
    plt.plot(xdata, ydata, ".", label="Data")
    optimizedParameters, pcov = opt.curve_fit(func, np.asarray(xdata), np.asarray(ydata))

    xinterp   = np.linspace(min(xdata), max(xdata), 100)
    fitinterp = [func(x, *optimizedParameters) for x in xinterp]
    plt.plot(xinterp, fitinterp, label="fit")
    plt.legend()
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print('The best fit parameters are: ', [p for p in optimizedParameters])
    return optimizedParameters
```

Import libraries

Fitting data with equations

- **Today we will use our first *Juypter notebook* to analyze data with a computer**

- **This is a simple python script to fit some data with a function and plot the results**

In [33]:

```
'''
This code is used to set up the fitting library.  Run this block without modification.
'''

%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as opt

'''
Arguments:
  - func:   the callable function used to fit the data.
            First argument must be the indpendent variable, rest of arguments are free parameters
  - xdata:  list of x axis coordinates of the data
  - ydata:  list of y axis coordinates of the data
  - xlabel: the title to put on the x axis
  - ylabel: the title to put on the y axis

Returns:
  - optimizedParameters: a list of the best fit values of the function parameters
'''
def fit(func, xdata, ydata, xlabel='', ylabel=''):
    plt.plot(xdata, ydata, ".", label="Data")
    optimizedParameters, pcov = opt.curve_fit(func, np.asarray(xdata), np.asarray(ydata))

    xinterp   = np.linspace(min(xdata), max(xdata), 100)
    fitinterp = [func(x, *optimizedParameters) for x in xinterp]
    plt.plot(xinterp, fitinterp, label="fit")
    plt.legend()
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print('The best fit parameters are: ', [p for p in optimizedParameters])
    return optimizedParameters
```

Import libraries

Run the “Curve fit” function

Fitting data with equations

- **Today we will use our first *Juypter notebook* to analyze data with a computer**

- **This is a simple python script to fit some data with a function and plot the results**

In [33]:

```
'''
This code is used to set up the fitting library.  Run this block without modification.
'''

%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as opt

'''
Arguments:
  - func:   the callable function used to fit the data.
            First argument must be the indpendent variable, rest of arguments are free parameters
  - xdata:  list of x axis coordinates of the data
  - ydata:  list of y axis coordinates of the data
  - xlabel: the title to put on the x axis
  - ylabel: the title to put on the y axis

Returns:
  - optimizedParameters: a list of the best fit values of the function parameters
'''
def fit(func, xdata, ydata, xlabel='', ylabel=''):
    plt.plot(xdata, ydata, ".", label="Data")
    optimizedParameters, pcov = opt.curve_fit(func, np.asarray(xdata), np.asarray(ydata))

    xinterp    = np.linspace(min(xdata), max(xdata), 100)
    fitinterp = [func(x, *optimizedParameters) for x in xinterp]
    plt.plot(xinterp, fitinterp, label="fit")
    plt.legend()
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print('The best fit parameters are: ', [p for p in optimizedParameters])
    return optimizedParameters
```

Import libraries

Run the “Curve fit” function

Make a nice plot

Fitting data with equations

- **Today we will use our first *Juypter notebook* to analyze data with a computer**

- **This is a simple python script to fit some data with a function and plot the results**

In [33]:

```
'''
This code is used to set up the fitting library.  Run this block without modification.
'''

%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as opt

'''
Arguments:
  - func:   the callable function used to fit the data.
            First argument must be the indpendent variable, rest of arguments are free parameters
  - xdata:  list of x axis coordinates of the data
  - ydata:  list of y axis coordinates of the data
  - xlabel: the title to put on the x axis
  - ylabel: the title to put on the y axis

Returns:
  - optimizedParameters: a list of the best fit values of the function parameters
'''
def fit(func, xdata, ydata, xlabel='', ylabel=''):
    plt.plot(xdata, ydata, ".", label="Data")
    optimizedParameters, pcov = opt.curve_fit(func, np.asarray(xdata), np.asarray(ydata))

    xinterp    = np.linspace(min(xdata), max(xdata), 100)
    fitinterp = [func(x, *optimizedParameters) for x in xinterp]
    plt.plot(xinterp, fitinterp, label="fit")
    plt.legend()
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print('The best fit parameters are: ', [p for p in optimizedParameters])
    return optimizedParameters
```

Import libraries

Run the “Curve fit” function

Make a nice plot

Return results

Fitting data with equations

- **Today we will use our first *Juypter notebook* to analyze data with a computer**
- **This is a simple python script to fit some data with a function and plot the results**

In [33]:

```
'''
This code is used to set up the fitting library.  Run this block without modification.
'''

%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import scipy.optimize as opt

'''
Arguments:
  - func:   the callable function used to fit the data.
            First argument must be the indpendent variable, rest of arguments are free parameters
  - xdata:  list of x axis coordinates of the data
  - ydata:  list of y axis coordinates of the data
  - xlabel: the title to put on the x axis
  - ylabel: the title to put on the y axis

Returns:
  - optimizedParameters: a list of the best fit values of the function parameters
'''
def fit(func, xdata, ydata, xlabel='', ylabel=''):
    plt.plot(xdata, ydata, ".", label="Data")
    optimizedParameters, pcov = opt.curve_fit(func, np.asarray(xdata), np.asarray(ydata))

    xinterp    = np.linspace(min(xdata), max(xdata), 100)
    fitinterp = [func(x, *optimizedParameters) for x in xinterp]
    plt.plot(xinterp, fitinterp, label="fit")
    plt.legend()
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print('The best fit parameters are: ', [p for p in optimizedParameters])
    return optimizedParameters
```

Import libraries

Always put detailed comments In your code so you remember What you meant later!

Run the “Curve fit” function

Make a nice plot

Return results

Fitting data with equations

- This is the result of running the code with the appropriate data and function from this lab.
- The parameter we are interested in is what voltage do we apply to completely stop the flow of current
- The code will automatically determine this value for you