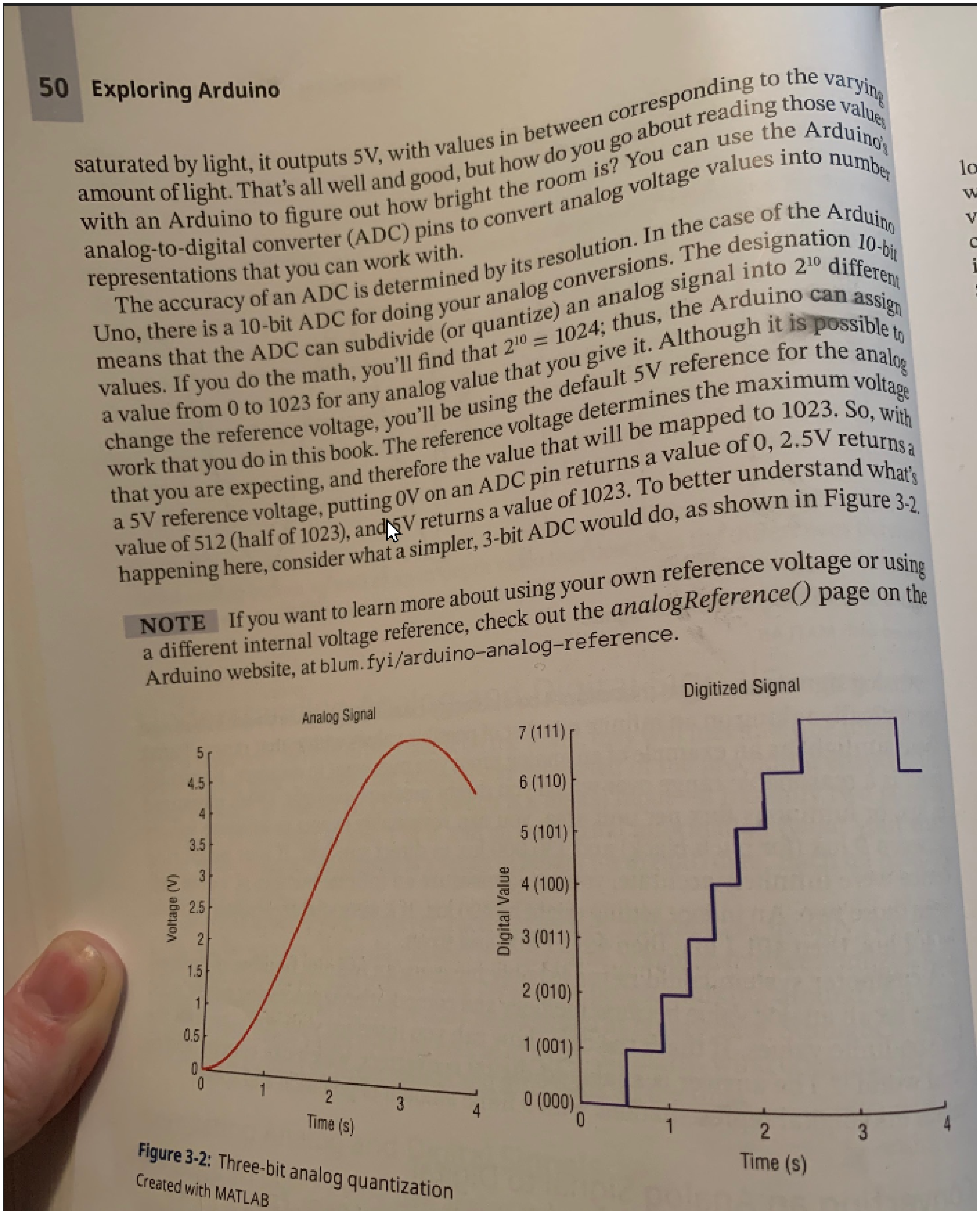
In [17]:

```python
# picture from Jeremy Blum's book "Exploring Arduino" 2nd edition Page 50
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://i.imgur.com/K6pJCwd.png")
```

Out[17]:



50 **Exploring Arduino**

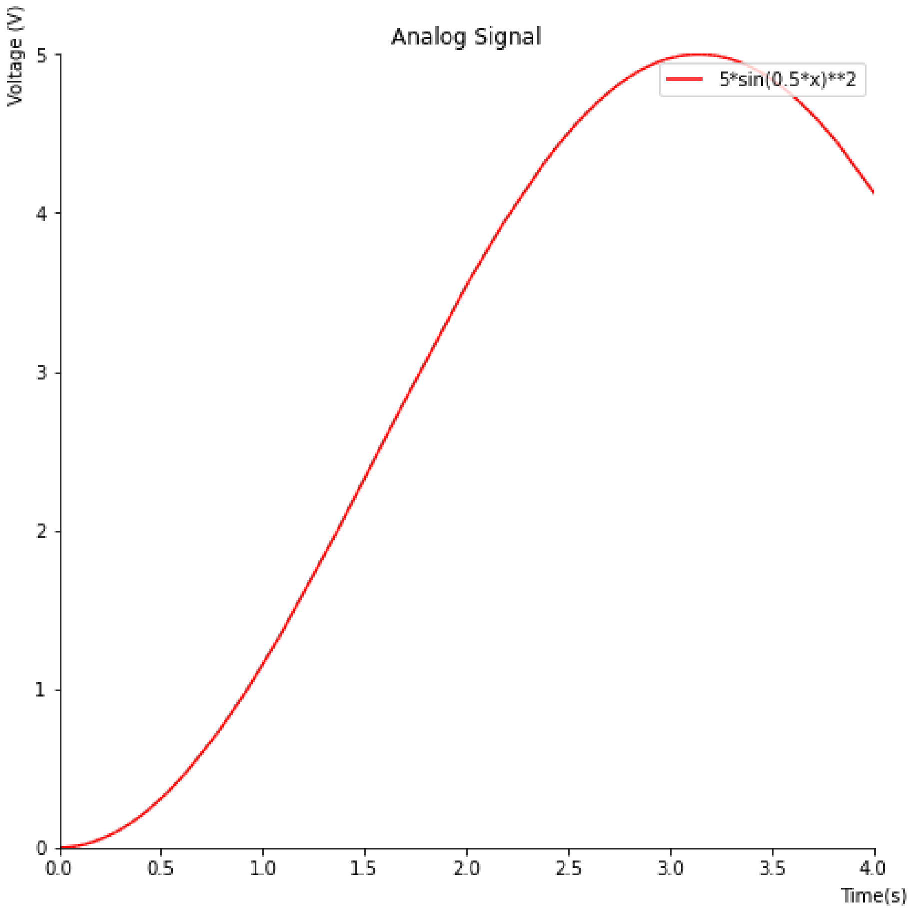
saturated by light, it outputs 5V, with values in between corresponding to the varying amount of light. That's all well and good, but how do you go about reading those values with an Arduino to figure out how bright the room is? You can use the Arduino's analog-to-digital converter (ADC) pins to convert analog voltage values into number representations that you can work with.

The accuracy of an ADC is determined by its resolution. In the case of the Arduino Uno, there is a 10-bit ADC for doing your analog conversions. The designation 10-bit means that the ADC can subdivide (or quantize) an analog signal into 2^{10} different values. If you do the math, you'll find that $2^{10} = 1024$; thus, the Arduino can assign a value from 0 to 1023 for any analog value that you give it. Although it is possible to change the reference voltage, you'll be using the default 5V reference for the analog work that you do in this book. The reference voltage determines the maximum voltage that you are expecting, and therefore the value that will be mapped to 1023. So, with a 5V reference voltage, putting 0V on an ADC pin returns a value of 0, 2.5V returns a value of 512 (half of 1023), and 5V returns a value of 1023. To better understand what's happening here, consider what a simpler, 3-bit ADC would do, as shown in Figure 3-2.

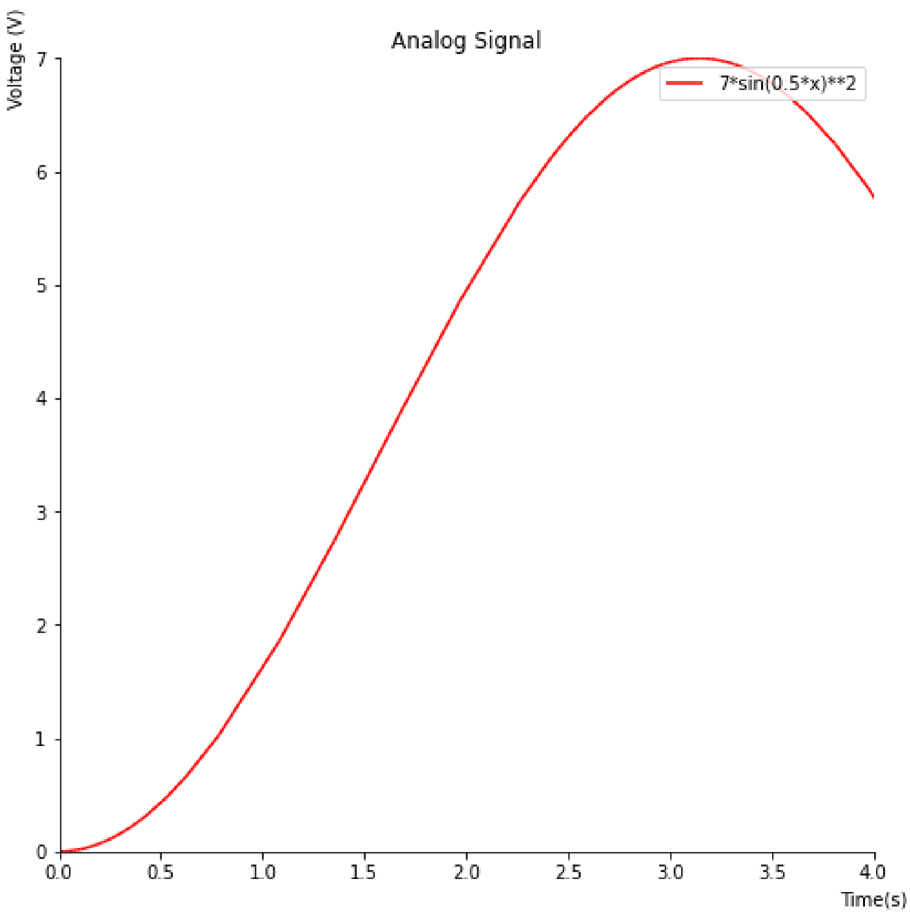
NOTE If you want to learn more about using your own reference voltage or using a different internal voltage reference, check out the *analogReference()* page on the Arduino website, at blum.fyi/arduino-analog-reference.

Figure 3-2: Three-bit analog quantization
Created with MATLAB

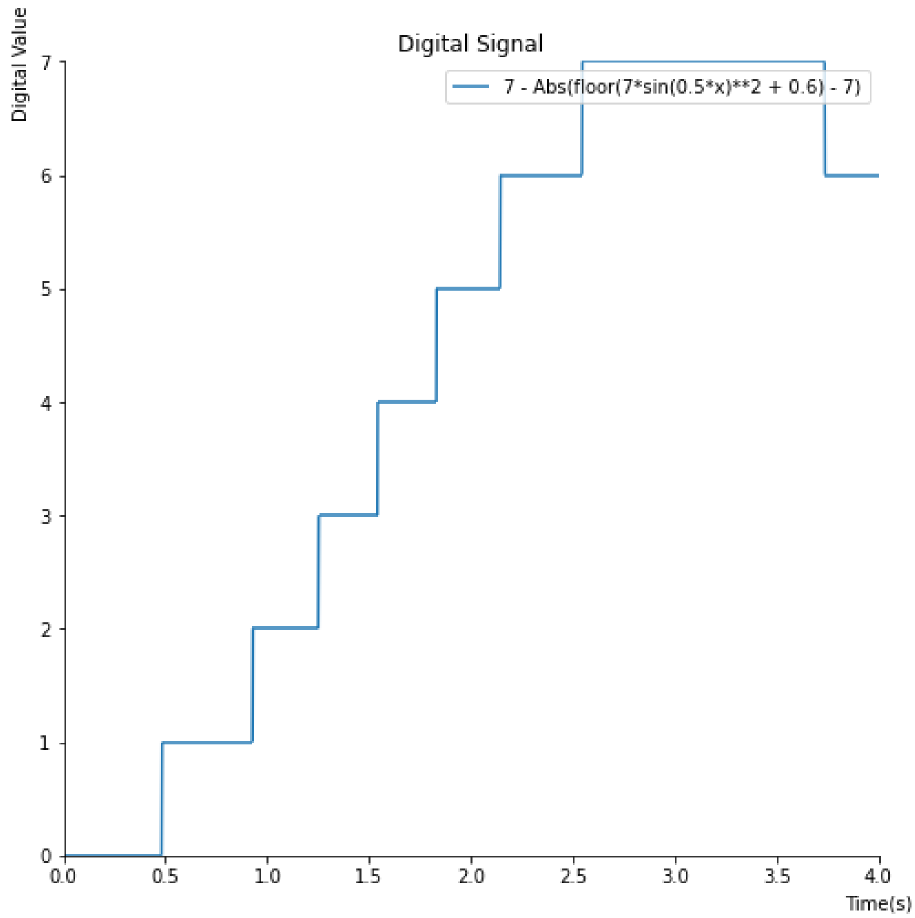
In [13]:
```
# through trial and error we find that 5*sin(0.5*x)**2 is very close to Blum's g
# the problem is that I could not figure out how to get the same amount of steps
# But with 7*sin(0.5*x)**2 then making the digital signal graph starts looking l
from sympy import *
x = symbols('x')
eq0 = 5*sin(0.5*x)**2
p = plot(eq0,legend = True,xlim = (0,4),ylim = (0,5),title="Analog Signal", xlab
p.show()
```



In [14]:
```
# What is unique here to my work was reverse engineering the graphs of Jeremy Bl
#
# Languages: Python Object Oriented Programming Language, Sympy (symbolic mathem
#       and Jupyter Notebook web-based Integrated Development Environment (IDE)
# https://docs.sympy.org/
# https://python.org/
# https://jupyter.org/
#
# referencing Jeremy Blum's "Exploring Arduino" - 2nd edition P.50 for the idea
# thank you Andrew F. Rich for help with digital function notation
eq = 7*sin(0.5*x)**2
p = plot(eq,legend = True,xlim = (0,4),ylim = (0,7),title="Analog Signal", xlabe
p.show()

# here we continually guessed values for some sine equation to look similar to P
# The differing aspect ratios of the images can make the sine wave function look
#
# notice our y-axis is different (mine has 7 versus  versus Blum's 5)
#     this is because I don't know how to get digital function to graph the 7 or
```



In [8]:
```
# notice here we use the previous 7*sin(0.5*x)**2 (variable: eq) as an input for
eq2 =  ((7 - Abs(floor((eq+0.6)) - 7)))

p = plot(eq2,legend = True,xlim = (0,4),ylim = (0,7),title="Digital Signal", xla
p.show()

# here we also reverse engineered Blum's digital value graph approximately
```



In [ ]:
```
Digital Value of range [0,7] translated to 3-bit ADC would be
    7 == 111
    6 == 110
    5 == 101
    4 == 100
    3 == 011
    2 == 010
    1 == 001
    0 == 000

 also important to note that 3-bit ADC means the ADC can quantize analog signals
 so a 10-bit ADC can do 2**10 or 1024 representations. In these 3-bit graphs, yo
 from a 5 volt supply. (although I had to use 7 volts to graph Blum's digital si

 In my personal case using the Arduino Metro microcontroller,
     a 5V regulator can supply peak ~800mA as long as the die temp of the regula
     using https://www.rapidtables.com/calc/electric/ohms-law-calculator.html
     that means we have:

         Resistance (R) = 6.25 ohms
         Current (I) = 800 milliamps (mA)
         Voltage (V) = 5 Volts (V)
         Power (P) = 4 watts(W)
```

In [15]:
```
eq # Analog Signal Function
```

Out[15]: $7\sin^2(0.5x)$

In [16]:
```
eq2  # Digital Signal Function
```

Out[16]: $7 - \left|\left\lfloor 7\sin^2(0.5x) + 0.6\right\rfloor - 7\right|$

In [ ]:
```
in conclusion, with 5*sin(0.5*x)**2 I had trouble getting the stepwise function
but with 7*sin(0.5*x)**2 the digital graph looks more like Blum's. So in reverse
I was able to get either the Analog graph correct and Digital graph incorrect or
and digital graph correct.
```

In [ ]:
```
special thanks to Robin R Mitchell for ideas about low level binary machine tran

written by    Nicholas Caudill          NSCaudill2020@manchester.edu
```