PY371 eLab Final Project Proposal: Moon Phase "Projector"/Diagram

Khloe "Keke" Katende*
Boston University Physics Department,
3 Cummington Mall, Boston, MA 02215
(Dated: May 1, 2021)

Usage: Visual translation of moon phases.

Structure: I am planning to program and build circuit that determines from an PNG image the phase that the moon is in by detecting the how many pixels are lit up. For my output, I will have an arrangement of LEDs (Light-Emitting Diodes) that "translate" the image unto itself.

I. NECESSARY EQUIPMENT:

1x Arduino UNO – For programming the LED.
2x Breadboard – For wiring purposes.
1x circular LED matrix – For "visual translation".
1x Raspberry Pi – For communicating and programming the Arduino UNO.

A. Implementation:

At its simplest, this project should function in such a way that:

1. *Process:*

1. A PNG file will be uploaded to the Raspberry Pi, which will translate it to an achromatic array of pixels (i.e. a bitmap of a black and white image) the Arduino UNO can process.

2. The Arduino UNO will take the bitmap and run a separate code that tells the LED matrix how to light up. On a simpler vesrion with only 8 LEDs, only one will light up per moon phase.

a. Extra: If I am able to and I have the time, I will first try to vary the luminosity for the in-between phases of the moon (e.x. if it is not quite a full moon, but more than a waxing gibbous moon, the luminosity on the full moon light will be lower than average).

2. *Timeline:*

1. *Week 0:* Figure out how many photoresistors I will need, and contact my Python professor on ideas he would have to write code to process images/the amount of light in them.

2. *Week 1:* Start building the code. Begin programming the Raspberry Pi.
3. *Week 2:* Start building the circuit proper. Finish programming Raspberry Pi, and begin programming Arduino UNO.
4. *Week 3:* Finish preliminary circuit and make sure it is functional; troubleshoot if need be.
5. *Week 4:* Refining.

II. FIGURES, DIAGRAMS, AND CODE:

A. Figures and Diagrams:

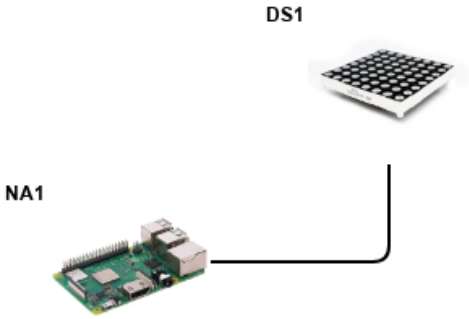

FIG. 1. Schematic drawing: Photo input → "Processed" by Raspberry Pi → LED display lights up as assigned by code

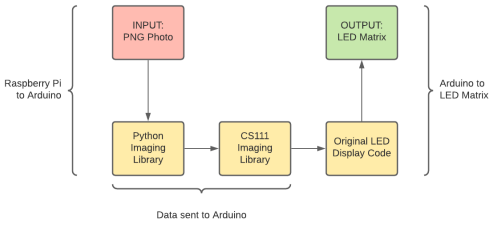

FIG. 2. Block diagram

* kkmen@bu.edu

B. Code Snippets:

```
#
#
# PY371: Electronic Lab for Students
#
# Final Project: Moon Diagram
# Part I: Raspberry Pi - Python 3
#

from hmcpng import *
import serial

# arduino = serial.Serial(port='COM4',
    baudrate=115200, timeout=.1)

# code provided by Prof. Sullivan for CS111

def brightness(pixel):
    """ takes a pixel (an [R, G, B] list) and
        returns a value between 0 and 255 that
        represents the brightness of that pixel.
    """
    red = pixel[0]
    green = pixel[1]
    blue = pixel[2]
    return (21*red + 72*green + 7*blue) // 100

def bw(pixels, threshold):
    """ Takes the 2-D list pixels containing pixels
        for an image, and that creates and returns
        a new 2-D list of pixels for an image that
        is a black-and-white version of the
        original image.
    inputs: an integer threshold between 0 and 255
        that should govern which pixels are turned
        white and which are turned black.
    """
    bwimg = blank_image(len(pixels),len(pixels[0]))

    for r in range(len(pixels)):
        for c in range(len(pixels[0])):
            if brightness(pixels[r][c]) > threshold:
                bwimg[r][c] = [255,255,255]
            else:
                bwimg[r][c] = [0,0,0]

    return bwimg

def LEDdisplay(pixels):
    """ Creates a new image from the original,
        changing color values from color to black
        and white based on brightness, then
        converts it into a byte array for
        processing by an Arduino UNO.
    """

    ser=serial.Serial("/dev/ttyACM0",9600)
    ser.baudrate=9600
    pixels = bw(pixels, min(brightness(pixel)))
    img = pixels.tobitmap()
    return img
```

```
//
// PY371: Electronic Lab for Students
//
// Final Project: Moon Diagram
// Part II: Arduino UNO - C for Arduino
//

const unsigned char PROGMEM img2[] =
{
  img2 = Serial.readBytes(256, img2, 256)
};

void drawBitmap(int x, int y, int sx, int sy,
    unsigned int *data)
{
 int tc = 0;
 for(int Y = 0; Y < sy; Y++)
 {
  for(int X = 0; X < sx; X++)
  {
   display.drawPixel(X+x, Y+y,
       pgm_read_word(&data[tc]));
   if(tc < (sx*sy)) tc++;
  }
 }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(
}

void loop() {
  // put your main code here, to run repeatedly:
  while (!Serial);
  Serial.println("OK");
  matrix.begin();
  matrix.drawBitmap(0, 0, img2, 256, 256, 0xFFFF);
}
```