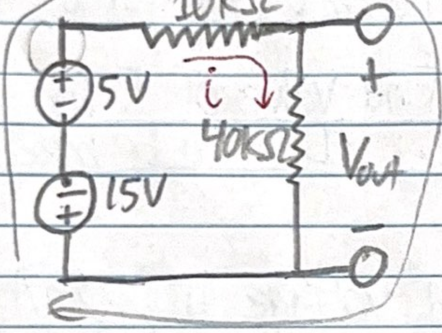
MECHATRONICS HW 1

Nathan Delos Santos

① Find: V_{out}



• KVL: $15V - 5V + i10k\Omega + i40k\Omega = 0$

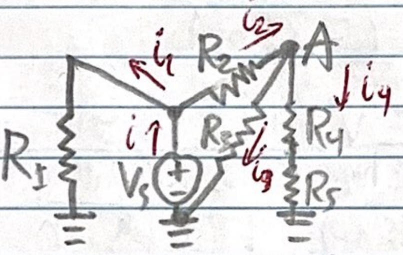
$10V = -50k\Omega \cdot i$

$i = -0.0002A = -0.2mA$

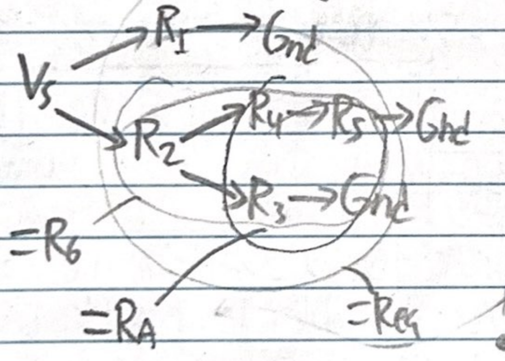
• Ohm's Law: $V_{out} = V_{40k\Omega} = i \cdot 40k\Omega$

$\boxed{V_{out} = -8V}$

② • $R_1 = 1k\Omega$ • $R_2 = 2k\Omega$
• $R_3 = 3k\Omega$ • $R_4 = 4k\Omega$
• $R_5 = 1k\Omega$ • $V_s = 10V$



Find: total equivalent resistance between V_s & gnd



• $R_{eq} = R_1 || R_6 \Rightarrow R_1 || (R_2 + R_3 || (R_4 + R_5))$

• $R_6 = R_2 + \dfrac{1}{\dfrac{1}{R_3} + \dfrac{1}{(R_4+R_5)}}$

$\Rightarrow R_6 = 3.875k\Omega$
$= (31/8)k\Omega$

• $R_{eq} = R_1 || R_6 = \dfrac{1}{\dfrac{1}{R_1} + \dfrac{1}{R_6}} \Rightarrow \boxed{R_{eq} = \frac{31}{39}k\Omega \simeq 0.795k\Omega}$

Find: V_A & current thru R_5

• Voltage division between: R_2 & $(R_3 || (R_4+R_5))$

• $V_s = V_{R2} + V_A \Rightarrow V_s = i_2 R_2 + V_A$
• $V_s = i_2(R_2 + R_A) \Rightarrow i_2 = \dfrac{V_s}{R_2 + R_A}$

☆ $R_A = R_3 || (R_4 + R_5) = \dfrac{1}{\frac{1}{3} + \frac{1}{4+1}}$

$R_A = \frac{15}{8}k\Omega = 1.875k\Omega$

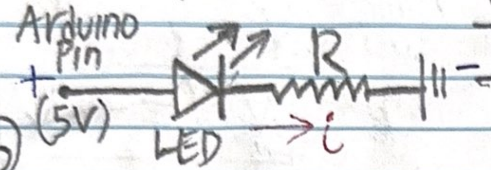
• Plugging in: $V_s = \dfrac{V_s}{R_2 + R_A} R_2 + V_A \Rightarrow V_s\left(1 - \dfrac{R_2}{R_2 + R_A}\right) = V_A \Rightarrow \boxed{V_A = V_s \dfrac{R_A}{R_2 + R_A} \Rightarrow V_A \simeq 4.84V}$

• Current thru R_5 is i_4

$V_A = i_4(R_4 + R_5) \Rightarrow i_4 = \dfrac{V_A}{R_4 + R_5} \Rightarrow \boxed{i_4 \simeq 0.9677 mA}$

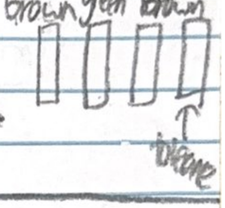
4) a) on next pg, forward voltage of red & yellow leds:
• red: 1.6–2.0V • yellow: 2.1–2.2V, so will use 2.0V for the problem, since is yellow

Find: value of R for yellow LED to be max brightness



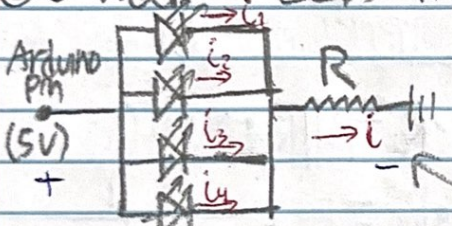
b)
• note, current at forward voltage is 20mA
• so $i = 20mA$

$5V = V_{led} + iR \Rightarrow 5V = 2.0V + 20mA(R) \Rightarrow R = 150\Omega$



• If use higher resistor, LED is dimmer. Lower, LED might burn

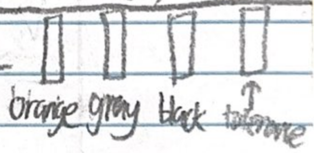
c) consider 4 LEDs in parallel



• $i_1 = i_2 = i_3 = i_4 = i_{led}$, so $i = 4 \cdot i_{led}$

• $5V = V_{led} + iR \Rightarrow 5V = 2.0V + 80mA(R)$

$R = 37.5\Omega \approx 38\Omega$



while it is best to power LEDs in parallel, connecting their common grounds to one resistor, like above, can be problematic, since the resistor might not be able to handle the current (more accurately, the power) going thru it

d) It is not advisable to connect 4 yellow leds in series if powered by an arduino pin. The arduino can only provide max 5V, and 4 yellow leds would require 8V drop. The leds wouldn't be fully powered

In addition, it's not good to wire lights in series. If one goes out, all go out, and it becomes hard to tell which one is busted

[1]“The Forward Voltages of Different LEDs | CircuitBread,” *The Forward Voltages of Different LEDs | CircuitBread*. https://www.circuitbread.com/ee-faq/the-forward-voltages-of-different-leds

[2]“LED - Basic Yellow 5mm - COM-09594 - SparkFun Electronics,” *www.sparkfun.com*. https://www.sparkfun.com/products/9594#:~:text=It%20has%20a%20typical%20forward

```
clc;
clear;
close all;

digitsDesired = 3; %%%enter digits here!
primes = primesUpToDigit(digitsDesired) %%%call it here!!!
sprintf("Sum of primes up to %d digits: %d",digitsDesired, sum(primes))

function [primes] = primesUpToDigit(digits)
    maxNum = 10^digits - 1; %for example, if want 3 digits, then get 10^3 -1 =
 1000 -1 = 999, the largest 3 digit numbner
    primes = [2,3]; %will get populated as the loop progresses
    startNumber = primes(end)+1; %will start collecting at the end of the
 primes list + 1, so in this case, at 3+1=4

    for i = startNumber:maxNum
        for ii = 1:length(primes)
            if mod(i,primes(ii)) == 0 %modulo to check for remainders of
 dividing i by primes(ii)
                %if remainder is 0, then that number is divisible, so break
                %loop, and skip this number
                break
            end
        end
        %if it reached the end of the list without breaking loop (for ii),
 then
        %that number was not found to be divisible by anything, so its prime
        %this works because you keep adding more and more numbers to the list,
        %so bigger numbers have more factors to check against
        if ii == length(primes)
            primes(end+1) = i;
        end
    end
    primes;
end
```

```
primes =

  Columns 1 through 13

     2     3     5     7    11    13    17    19    23    29    31    37    41

  Columns 14 through 26

    43    47    53    59    61    67    71    73    79    83    89    97   101

  Columns 27 through 39

   103   107   109   113   127   131   137   139   149   151   157   163   167

  Columns 40 through 52
```

```
   173   179   181   191   193   197   199   211   223   227   229   233   239

  Columns 53 through 65

   241   251   257   263   269   271   277   281   283   293   307   311   313

  Columns 66 through 78

   317   331   337   347   349   353   359   367   373   379   383   389   397

  Columns 79 through 91

   401   409   419   421   431   433   439   443   449   457   461   463   467

  Columns 92 through 104

   479   487   491   499   503   509   521   523   541   547   557   563   569

  Columns 105 through 117

   571   577   587   593   599   601   607   613   617   619   631   641   643

  Columns 118 through 130

   647   653   659   661   673   677   683   691   701   709   719   727   733

  Columns 131 through 143

   739   743   751   757   761   769   773   787   797   809   811   821   823

  Columns 144 through 156

   827   829   839   853   857   859   863   877   881   883   887   907   911

  Columns 157 through 168

   919   929   937   941   947   953   967   971   977   983   991   997


ans = 

    "Sum of primes up to 3 digits: 76127"
```

Published with MATLAB® R2022b

primeNumbersCollector.cpp

```cpp
#include <iostream>
#include <vector> //so can add elements to arrays at runtime
#include <cmath>

using namespace std;
int desiredDigits = 3;

vector<int> primeFinder(int digitCount){
    int maxNum = pow(10,digitCount) - 1;
    //for example, if want 3 digits, then get 10^3 -1 = 1000 -1 = 999, the largest 3 digit numbner
    vector<int> primes = {2,3};

    //cout << "maxNum: "<<maxNum << " \n";
    int ii; //need to declare this outside the loop so that stuff outside the loop can access it

    //sizeof(primes)/sizeof(primes[0]) is the length of the array, primes with plain c++ arrays
    //primes.size() is how it is done with vectors in c++
    //will start collecting at the end of the primes list + 1, so in this case, at 3+1=4
    for(int i = primes[primes.size() - 1] +1; i<=maxNum ; i++){
        //cout << "i = " << i << " \n";
        for(ii = 0;ii<primes.size();ii++){
            if(i % primes[ii] == 0){
                break;
                //modulo to check for remainders of dividing i by primes(ii)
                //if remainder is 0, then that number is divisible, so break
                //loop, and skip this number
            }
        }
        //cout << "checked, ii = " << ii << " , length of primes: " << primes.size() << " \n";
        //if it reached the end of the list without breaking loop (for ii), then
        //that number was not found to be divisible by anything, so its prime
        //this works because you keep adding more and more numbers to the list,
        //so bigger numbers have more factors to check against
        if(ii == primes.size()){
            primes.push_back(i);
        }
    }
    return primes;
}

int main(){
    int sumOfPrimes=0;
    vector<int> foundPrimes = primeFinder(desiredDigits);
    cout << "Primes Up To " << desiredDigits << " Digits: \n";
    for (const int &i : foundPrimes) {
        cout << i << " , ";
        sumOfPrimes += i;
    }
    cout << "\n";
    cout << "Sum Of Primes Up To " << desiredDigits << " Digits: "<< sumOfPrimes <<"\n";
    return 0;
}


```