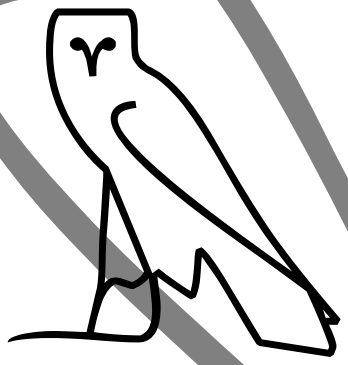
MYSTERIOUS OWL



U+13153

https://github.com/Mysterious-Owl/

https://www.youtube.com/channel/UCpkxxb7y9nIlUlft5GKTNsg

INDEX

INTRODUCTION

Calculator is very common device, everyone has used it in their life uncountable number of times and for many it's a lifeline. It's used to perform tedious mathematical calculations which are very difficult to perform by hands/mind. Nowadays calculator has become so advanced that they can also plot graphs, but they are so costly. Basic calculator is something that perform limited operations which is used by most like addition, subtraction, multiplication and division.

Here we tried to make a calculator which is able to perform various actions and it also support decimal numbers. We designed both hardware and software of the calculator, as both the things should be accurate and fast to provide accurate results in efficient time. It is based on microcontroller namely, ATmega328p, which is 8-bit microcontroller. It has OLED screen to show the numbers we type and to show the result. It can be implemented physically on a 10cm by 4.2 cm, which is small comparatively to present calculators, which make it handy and easy to carry. It has total 24 buttons, out of which 10 denotes numbers "0, 1, 2, 3, 4 … 9" and one button for decimal point "." and other for several other use, as stated.

It has total 7 operands, which can be performed, which are –

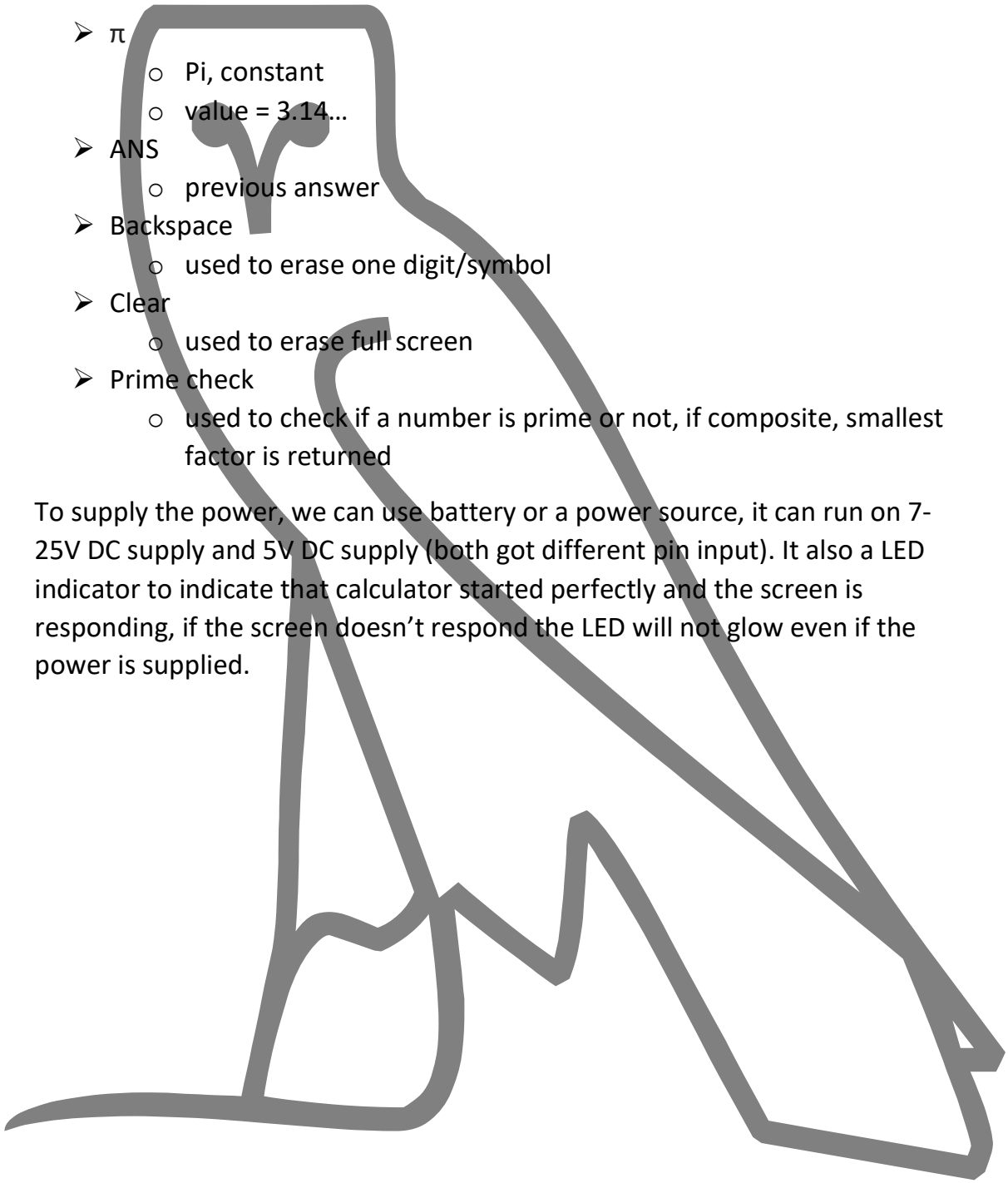
- \+ add
- \- subtract
- X multiply
- / divide
- ^ power function
- % modulus/remainder
- = equal to

It also has other features –

- e
  - Euler's number, constant
  - value = 2.71828….

- π
  - Pi, constant
  - value = 3.14…
- ANS
  - previous answer
- Backspace
  - used to erase one digit/symbol
- Clear
  - used to erase full screen
- Prime check
  - used to check if a number is prime or not, if composite, smallest factor is returned

To supply the power, we can use battery or a power source, it can run on 7-25V DC supply and 5V DC supply (both got different pin input). It also a LED indicator to indicate that calculator started perfectly and the screen is responding, if the screen doesn't respond the LED will not glow even if the power is supplied.



HARDWARE USED

Main working unit

1. OLED screen
   - 128*64 pixel
   - 0.96 inch
   - I^2C communication protocol used
   - Monochrome
   - 4 pins – SDA, SCL, VCC, GND
   - SSD1306 controller

2. ATmega328p microcontroller
   - 28 pin, DIP package
   - 16 MHz external clock
   - Arduino bootloader
   - 4.5 – 5.5 V, operating voltage
   - 8-bit microcontroller

3. Crystal
   - 16MHz
   - 2 pins

4. Ceramic Capacitor (2 units)
   - 22pF
   - 2 pins

5. Resistance
   - 10k Ω
   - 2 pins

6. Push to on switch (24 units)
   - 4 pins
   - When not pressed, 2 pins are shorted on both side
   - When pressed, all 4 pins are shorted

*For all components, unless otherwise stated, only 1 unit is used.

Indicator unit

7. LED
    - Any colour
    - 2 pins, anode and cathode

8. Resistance
    - 100 Ω
    - 2 pins

Power unit

9. IC7805
    - Voltage regulator IC
    - Input range 7-25V
    - Constant output 5V
    - 3 pins, Input, Common, Output

10. Electrolytic capacitor (2 units)
    - 10µF
    - 2 pins, positive, negative

Components required to make it physical, in simulation they are not required

11. Connectors
    - Used to connect jumper wires

12. PCB
    - Size 4.2cm * 10cm
    - Single side

*For all components, unless otherwise stated, only 1 unit is used.

SOFTWARE USED

1) Arduino IDE for code compilation and HEX file generation.
2) Proteus for simulation, PCB designing and 3D model generation.

Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards and microcontrollers.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

With the rising popularity of Arduino as a software platform, other vendors started to implement custom open-source compilers and tools (cores) that can build and upload sketches to other microcontrollers that are not supported by Arduino's official line of microcontrollers.

Here, we are using this software for code compilation and HEX file generation.

Proteus

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. It is a Windows application for schematic capture, simulation, and PCB (Printed Circuit Board) layout design. It can be purchased (it's a paid software) in many configurations, depending on the size of designs being produced and the requirements for microcontroller simulation.

- Schematic Capture

Schematic capture in the Proteus Design Suite is used for both the simulation of designs and as the design phase of a PCB layout project. It is therefore a core component and is included with all product configurations.

- Microcontroller Simulation

The micro-controller simulation in Proteus works by applying either a hex file or a debug file to the microcontroller part on the schematic. It is then co-simulated along with any analog and digital electronics connected to it. This enables its use in a broad spectrum of project prototyping in areas such as motor control, temperature control and user interface design. It also finds use in the general hobbyist community and, since no hardware is required, is convenient to use as a training or teaching tool.

- PCB Design

The PCB Layout module is automatically given connectivity information in the form of a netlist from the schematic capture module. It applies this information, together with the user specified design rules and various design automation tools, to assist with error free board design. PCB's of up to 16 copper layers can be produced with design size limited by product configuration.

- 3D Verification

The 3D Viewer module allows the board under development to be viewed in 3D together with a semi-transparent height plane that represents the boards enclosure. STEP output can then be used to transfer to mechanical CAD software such as Solidworks or Autodesk for accurate mounting and positioning of the board.

WORKING PRINCIPLE

This project is completely based on MCU, which is ATmega328p, here. MCU means “Microcontroller Unit”. It takes the input from the buttons, which is status of the button and it stores the entered data and accordingly show it on the screen after processing it. ATmega328 has many GPIO pins but we are using 10 for keys and 2 for OLED and one for indicator LED.

The matrix keypad design is applied in making the button of the calculator. It allows us to use minimum GPIO pins and gives better and accurate result. We also are not required to use any additional hardware. We have total of 24 buttons, 6 rows and 4 columns, thereby requiring only 10 pins to communicate. This design is most widely used design nowadays due to least requirement of pins. It is used in keyboard of computer.

For display we used an OLED display with pixel size of 128*64. OLED display is brighter and takes less power compared to LCD displays. In our use case we don’t require colour so we are using a monochrome display. To communicate with MCU we are using I^2C protocol, which consists of 2 lines, SDA (serial data) and SCL (serial clock). OLED is using SSD1306 driver to communicate with I^2C protocol. I^2C is serial communication protocol and every device in this connection has got its unique address, by which device is identified and therefore giving us ability to connect many devices (multi master and multi slave) from only 2 lines. SDA line is used to send the serial data, first address is sent over the line so that the all the devices can know that this data is meant for them or not. SCL is used to send the clock pulse, it’s a synchronous protocol, therefore it can’t be used in communications over long distance. Sometimes, display connection fail, due to many factors, so to handle that we have one LED, which will be turned on if the display is initialized successfully.

For power we are using IC7805 which is a voltage regulator IC, which can provide a constant voltage of 5V, when the input range is 7-25V, apart from this we can also directly connect a 5V regulated power supply via a different connector. So, we have total 3 pins for power input and we can use either 2. 3

pins are V1, GND and V2. V1 and GND are for 7-25V and V2 and GND for 5V supply, we also used 10µF capacitor to damp the disturbances in the voltage flow, if any. MCU and OLED both needs 5V regulated supply, that's why we need 5V supply.

Now the heart of the project, that is MCU. It's an 8-bit MCU, 28-pin DIP (Dual inline package) package, with clock speed of 16MHz, which is relatively slow from present day computer which is 2GHz, which is almost 125 times faster than our MCU, but it also cost too much. So, considering our use case and computations requirement 16MHz is sufficient. 16Mhz clock is fastest speed which can be provided by ATmega328p, if we want more we have to change our hardware. In ATmega328p, "p" stands for picopower implying it consumes very less power making it very power efficient. We added a crystal of 16MHz with two capacitors, which is working as its external clock, otherwise we have to use 8MHz internal clock, which will take twice as much time as it takes now. We ae using Arduino bootloader in the MCU, which allows us to code in Arduino language and use Arduino IDE to compile code and generate HEX files. Currently we made a code of approx. 600 lines, deploying 7 operator and 6 other features. For I^2C communication, this MCU has internal support, which makes it very compatible with OLED we are using.

Code of this project can be divided in many parts –

- Importing required libraries
- Initialising variables to store data
- Initialising hardware
- Checking if button is pressed
- Store the entered data
- Process the result if "=" is pressed and display the result
- Error correction code
- Constants value
- Prime check

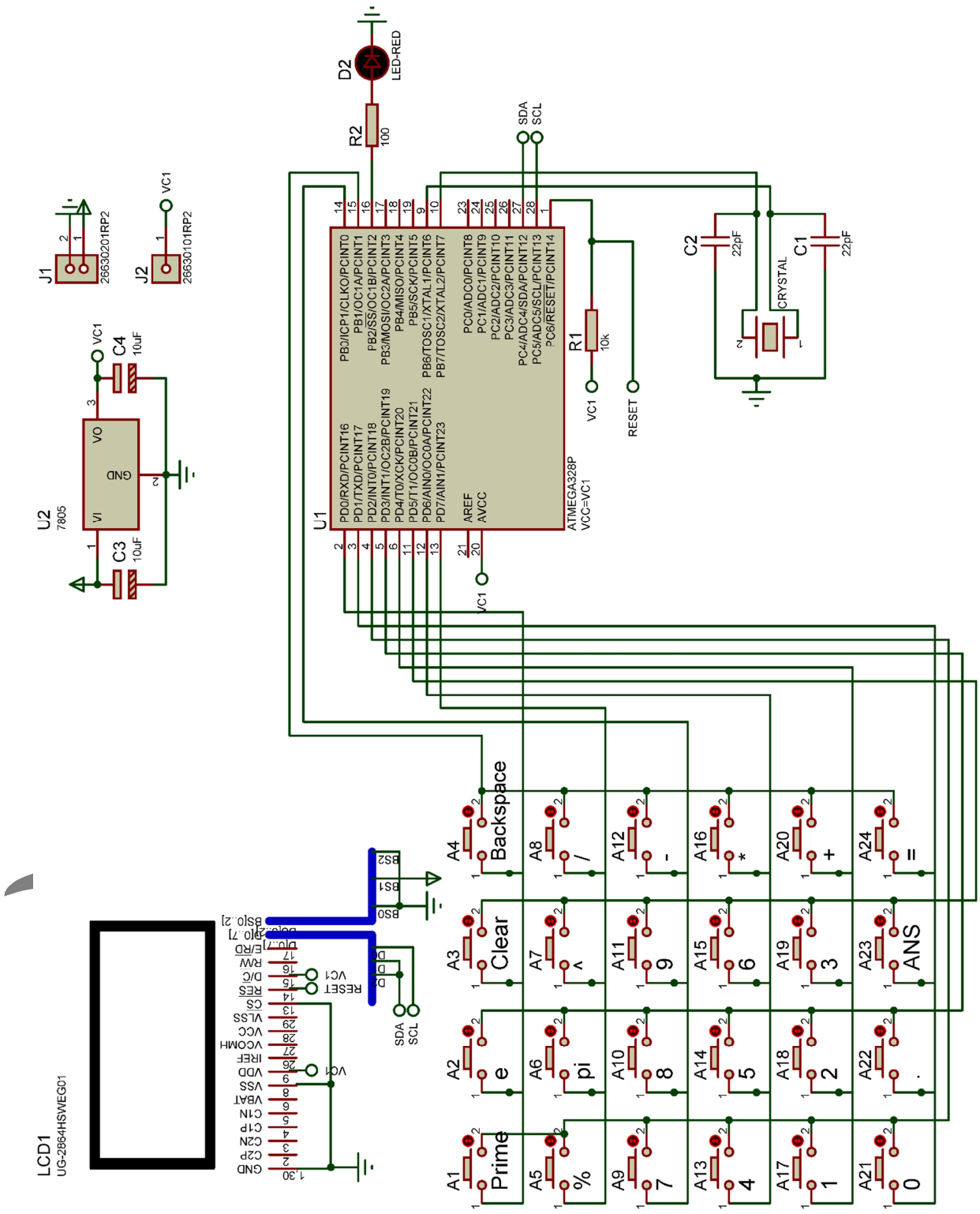
The code also has many flags, to tell if specific action is performed like for if any operator key is pressed directly without any number before it, so it will automatically take the previous answer as first operant, if any operator key is pressed then if digit key is pressed so to store the new digit as another

number. For constants we are using the constant provided a library, which has very high accuracy.

IDE is “Integrated Development Environment”, here we are using Arduino IDE, since we are using Arduino bootloader. We can upload the code to our MCU by using USB to TTL converter. For simulation purpose we are using HEX file generated by IDE and then upload it to our MCU. Arduino IDE lets us use many libraries easily and tells us about syntax error, if any. Arduino IDE is an open-source software.

Proteus is the software used for simulation. It’s a paid software made by “Labcenter Electronics Ltd”. We are using this software to simulate our project. To simulate we have to make the schematic diagram in the software and upload the code, if any. If any component is not there in its components library then either we can import from other sources or we can create our own component. We can create component with schematic design and also we have an option to design a PCB package for it and also provide a 3D model of the component. We can also make PCB layout of the schematic diagram, here in our project we placed the components by ourselves, but we can also use auto placer and auto router. We can export the PCB layout by exporting GERBER files. We can also see 3D view of our PCB layout, of the components whose 3D model is available, if not we can provide it from outside too.

SCHEMATIC DIAGRAM



SOURCE CODE

Download code from https://github.com/Mysterious-Owl/calculator/blob/main/Calculator.ino

(its uploaded by one of the members of group)

```
#include <Keypad.h>
#include <Wire.h>
#include <math.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET     -1

#define max_size  15
#define ROWS  6
#define COLS  4

char keys[ROWS][COLS] = {
  {'P','E','C','B'},
  {'%','I','^','/'},
  {'7','8','9','-'},
  {'4','5','6','*'},
  {'1','2','3','+'},
  {'0','.','A','='}
};
byte rowPins[ROWS] = {0,7,8,6,4,1}, colPins[COLS] = {2,3,5,9};
double num[max_size]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int op[max_size] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    state[max_size] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double ans=0;
int n=0,o=0,i=0,d=0;
String numa="";
bool fla=1,ans1=0,flad=0,flai=0;
char key;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
Keypad customKeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

```

```
35. void clearall();
36.
37. void setup()
38. {
39.   if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3D)) {
40.       for(;;);
41.   }
42.   pinMode(10,OUTPUT);
43.   digitalWrite(10,HIGH);
44.   display.setTextSize(2);
45.   display.setTextColor(SSD1306_WHITE);
46.   display.setCursor(0, 0);
47.   display.clearDisplay();
48.   display.write("Calculator");
49.   display.display();
50.   display.clearDisplay();
51.   display.setTextSize(2);
52.   display.setCursor(0, 0);
53. }
54.
55. void loop()
56. {
57.   key = customKeypad.getKey();
58.   switch(key)
59.       {
60.       case '0' ... '9':
61.           display.write(key);
62.           display.display();
63.           if(fla)   n++,fla=0;
64.           if(flad)  d++;
65.           num[n-1] = num[n-1]*10 + (key - '0');
66.           ans1=0;
67.           break;
68.
69.         case '+':
70.           if(ans1){
71.               display.write("Ans");
72.               display.display();
73.               num[n]=ans;
74.               n++;
75.               ans1=0;
76.           }
77.           if(flad){
78.             state[n-1]=d;
79.             num[n-1]=num[n-1]/pow(10,d);
80.             d=0;
81.             flad=0;
82.           }
```

```
83.           display.write(key);
84.           display.display();
85.           op[o] = 1;
86.           o++;
87.           fla=1;
88.           break;
89.
90.         case '-':
91.           if(ans1){
92.               display.write("Ans");
93.               display.display();
94.               num[n]=ans;
95.               n++;
96.               ans1=0;
97.           }
98.           if(flad){
99.             state[n-1]=d;
100.              num[n-1]=num[n-1]/pow(10,d);
101.              d=0;
102.              flad=0;
103.            }
104.            display.write(key);
105.            display.display();
106.            op[o] = 2;
107.            o++;
108.            fla=1;
109.            break;
110.
111.          case '*':
112.            if(ans1){
113.                state[n-1]=d;
114.                d=0;
115.                display.write("Ans");
116.                display.display();
117.                num[n]=ans;
118.                n++;
119.                ans1=0;
120.            }
121.            if(flad){
122.              state[n-1]=d;
123.              num[n-1]=num[n-1]/pow(10,d);
124.              d=0;
125.              flad=0;
126.            }
127.            display.print("X");
128.            display.display();
129.            op[o] = 3;
130.            o++;
```

```
131.             fla=1;
132.             break;
133.
134.           case '^' :
135.             if(ans1){
136.                 display.write("Ans");
137.                 display.display();
138.                 num[n]=ans;
139.                 n++;
140.                 ans1=0;
141.             }
142.             if(flad){
143.               state[n-1]=d;
144.               num[n-1]=num[n-1]/pow(10,d);
145.               d=0;
146.               flad=0;
147.             }
148.             display.write(key);
149.             display.display();
150.             op[o] = 4;
151.             o++;
152.             fla=1;
153.             break;
154.
155.           case '/' :
156.             if(ans1){
157.                 display.write("Ans");
158.                 display.display();
159.                 num[n]=ans;
160.                 n++;
161.                 ans1=0;
162.             }
163.             if(flad){
164.               state[n-1]=d;
165.               num[n-1]=num[n-1]/pow(10,d);
166.               d=0;
167.               flad=0;
168.             }
169.             display.write(key);
170.             display.display();
171.             op[o] = 5;
172.             o++;
173.             fla=1;
174.             break;
175.
176.           case '%' :
177.             if(ans1){
178.                 state[n-1]=d;
```

```
179.                d=0;
180.                display.write("Ans");
181.                display.display();
182.                num[n]=ans;
183.                n++;
184.                ans1=0;
185.              }
186.              if(flad){
187.                state[n-1]=d;
188.                num[n-1]=num[n-1]/pow(10,d);
189.                d=0;
190.                flad=0;
191.              }
192.              display.write(key);
193.              display.display();
194.              op[o] = 6;
195.              o++;
196.              fla=1;
197.              break;
198.
199.            case '=':
200.              display.print("  =");
201.              display.display();
202.              display.drawLine( 0,  46,  128, 46,
    SSD1306_WHITE );
203.              display.setCursor(0,48);
204.              if(n==0 && o==0){
205.                display.print("Enter something");
206.                display.display();
207.                goto againe;
208.              }
209.              if(o!=n-1  || flai){
210.                display.write("Invalid syntax");
211.                display.display();
212.                goto againe;
213.              }
214.              if(flad){
215.                state[n-1]=d;
216.                num[n-1]=num[n-1]/pow(10,d);
217.                d=0;
218.                flad=0;
219.              }
220.              i=o;
221.              for(;;){
222.                    if(i==0)  break;
223.                    else{
224.                      int max=0, ind=-1;
225.                      for(int h=0;h<i;h++){
```

```
226.                         if(op[h]>max){
227.                           max=op[h];
228.                           ind=h;
229.                         }
230.                       }
231.                       switch(op[ind]){
232.
233.                         case 1:
234.                           num[ind]=num[ind]+num[ind+1];
235.                           break;
236.
237.                         case 2:
238.                           num[ind]=num[ind]-num[ind+1];
239.                           break;
240.
241.                         case 3:
242.                           num[ind]=num[ind]*num[ind+1];
243.                           break;
244.
245.                         case 4:
246.
    num[ind]=pow(num[ind],num[ind+1]);
247.                           break;
248.
249.                         case 5:
250.                           if(num[ind+1]==0){
251.                             display.write("Wrong
    Input");
252.                             display.display();
253.                             goto againe;
254.                           }
255.                           num[ind]=num[ind]/num[ind+1];
256.                           break;
257.
258.                         case 6:
259.                           if(num[ind+1]==0){
260.                             display.write("Wrong
    Input");
261.                             display.display();
262.                             goto againe;
263.                           }
264.
    num[ind]=int(num[ind])%int(num[ind+1]);
265.                           break;
266.                       }
267.                       for(int h=ind+1;h<i+1;h++){
268.                         op[h-1]=op[h];
269.                         num[h]=num[h+1];
```

```
270.                     }
271.                     if(i==6){
272.                       op[5]=0;
273.                       num[5]=0;
274.                     }
275.                     i--;
276.                 }
277.               }
278.               numa=String(num[0],5);
279.               display.println(numa);
280.               display.display();
281.               ans=num[0];
282.               againe:
283.               clearall();
284.               break;
285.
286.             case 'C':
287.               clearall();
288.               display.display();
289.               break;
290.
291.             case 'A':
292.               display.write("Ans");
293.               display.display();
294.               num[n]=ans;
295.               state[n]=-2;
296.               ans1=0;
297.               n++;
298.               break;
299.
300.             case 'P':
301.               clearall();
302.               display.setTextSize(2);
303.               display.println("Is Prime?");
304.               display.drawLine( 0,  17,  128, 17,
    SSD1306_WHITE );
305.               display.display();
306.               display.setCursor(0,21);
307.               enumber();
308.               display.setTextSize(2);
309.               clearall();
310.               break;
311.
312.              case 'E':
313.               display.print("e");
314.               display.display();
315.               num[n]=M_E;
316.               state[n]=-3;
```

```
317.             n++;
318.             ans1=0;
319.             break;
320.
321.           case 'I':
322.             display.print("pi");
323.             display.display();
324.             num[n]=M_PI;
325.             state[n]=-4;
326.             n++;
327.             ans1=0;
328.             break;
329.
330.           case '.':
331.             if(flad)  flai=1;
332.             flad=1;
333.             d=0;
334.             display.write(key);
335.             display.display();
336.             break;
337.
338.           case 'B':
339.               if(flad){
340.                  state[n-1]=d;
341.                  num[n-1]=num[n-1]/pow(10,d);
342.                  d=0;
343.                  flad=0;
344.                }
345.                display.clearDisplay();
346.                display.setCursor(0, 0);
347.                if(o>=n){
348.                  int k=0;
349.                  o--;
350.                  op[o]=0;
351.                  for(k=0;k<n;k++){
352.                    switch(state[k]){
353.                      case 0:
354.                        numa=String(num[k],0);
355.                        display.print(numa);
356.                        break;
357.                      case -2:
358.                        display.print("Ans");
359.                        break;
360.                      case -3:
361.                        display.print("e");
362.                        break;
363.                      case -4:
364.                        display.print("pi");
```

```
365.                         break;
366.                       default:
367.                         numa=String(num[k],state[k]);
368.                         display.print(numa);
369.                         break;
370.                     }
371.                     switch(op[k]){
372.                       case 1:
373.                         display.write("+");
374.                         break;
375.
376.                       case 2:
377.                         display.write("-");
378.                         break;
379.
380.                       case 3:
381.                         display.write("X");
382.                         break;
383.
384.                       case 4:
385.                         display.write("^");
386.                         break;
387.
388.                       case 5:
389.                         display.write("/");
390.                         break;
391.
392.                       case 6:
393.                         display.write("%");
394.                         break;
395.                     }
396.                   }
397.                   for(;k<o;k++){
398.                     switch(op[k]){
399.                       case 1:
400.                         display.write("+");
401.                         break;
402.
403.                       case 2:
404.                         display.write("-");
405.                         break;
406.
407.                       case 3:
408.                         display.write("X");
409.                         break;
410.
411.                       case 4:
412.                         display.write("^");
```

```
413.                         break;
414.
415.                       case 5:
416.                         display.write("/");
417.                         break;
418.
419.                       case 6:
420.                         display.write("%");
421.                         break;
422.                     }
423.                   }
424.                 }
425.                 else{
426.                   n--;
427.                   for(int k=0;k<n;k++){
428.                     switch(state[k]){
429.                       case 0:
430.                         numa=String(num[k],0);
431.                         display.print(numa);
432.                         break;
433.                       case -2:
434.                         display.print("Ans");
435.                         break;
436.                       case -3:
437.                         display.print("e");
438.                         break;
439.                       case -4:
440.                         display.print("pi");
441.                         break;
442.                       default:
443.                         numa=String(num[k],state[k]);
444.                         display.print(numa);
445.                         break;
446.                     }
447.                     switch(op[k]){
448.                       case 1:
449.                         display.write("+");
450.                         break;
451.
452.                       case 2:
453.                         display.write("-");
454.                         break;
455.
456.                       case 3:
457.                         display.write("X");
458.                         break;
459.
460.                       case 4:
```

```
461.                         display.write("^");
462.                         break;
463.
464.                       case 5:
465.                         display.write("/");
466.                         break;
467.
468.                       case 6:
469.                         display.write("%");
470.                         break;
471.                     }
472.                   }
473.                   if(state[n]==0){
474.                     numa=String(num[n],0);
475.
    numa=numa.substring(0,numa.length()-1);
476.                     num[n]=numa.toDouble();
477.                     if(num[n]!=0){
478.                       display.print(numa);
479.                     }
480.                     else{
481.                       fla=1;
482.                       n--;
483.                     }
484.                   }
485.                   else if(state[n]>0){
486.                     numa=String(num[n],state[n]);
487.                     state[n]--;
488.                     d=state[n];
489.
    numa=numa.substring(0,numa.length()-1);
490.                     num[n]=numa.toDouble()*pow(10,d);
491.                     display.print(numa);
492.                     if(state[n]!=-1) flad=1;
493.                   }
494.                   else{
495.                     num[n]=0;
496.                     state[n]=0;
497.                     n--;
498.                     fla=1;
499.                   }
500.                   n++;
501.                 }
502.                 display.display();
503.                 break;
504.       }
505.   }
506.
```

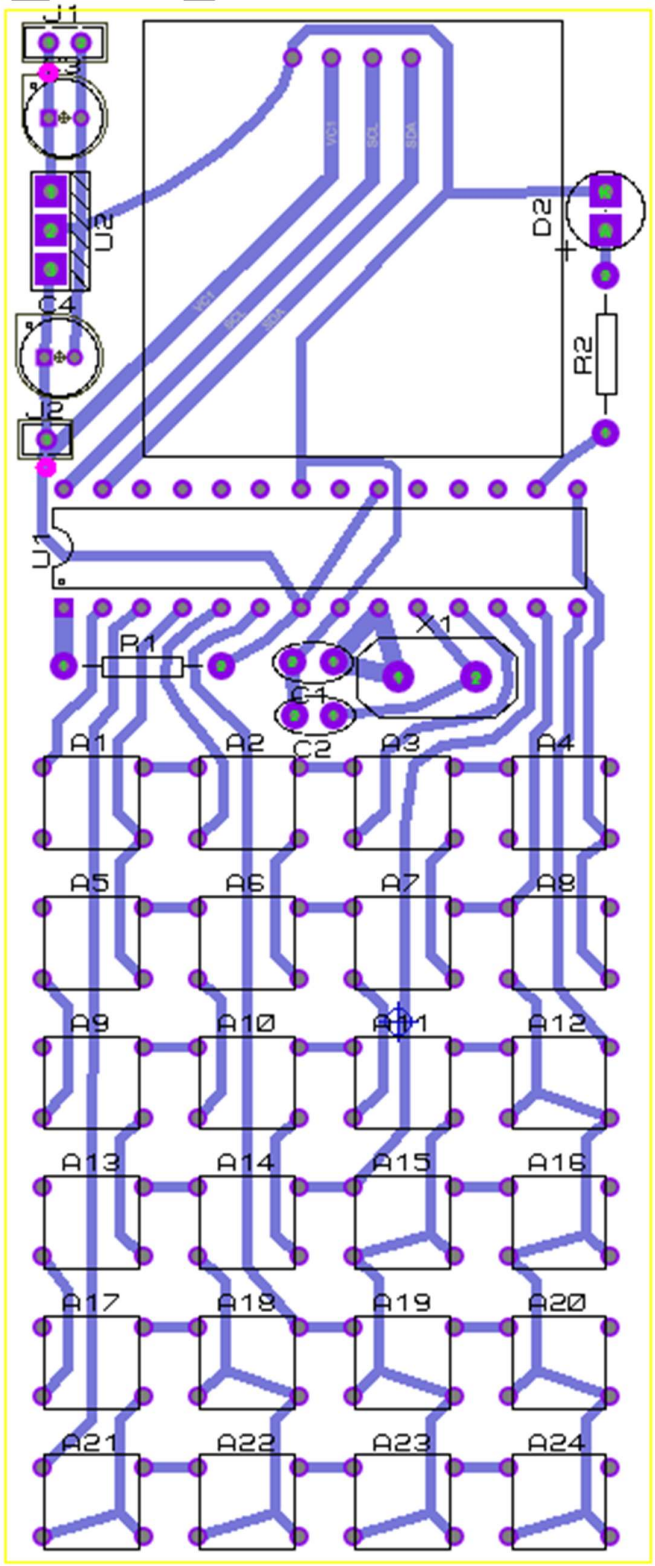
```
507.
508.  void clearall(){
509.    for(int jl=0;jl<max_size;jl++){
510.      num[jl]=0;
511.      op[jl]=0;
512.      state[jl]=0;
513.    }
514.    i=0;
515.    o=0;
516.    ans1=1;
517.    n=0;
518.    fla=1;
519.    flai=0;
520.    flad=0;
521.    d=0;
522.    display.setCursor(0,0);
523.    display.clearDisplay();
524.  }
525.
526.
527.  void enumber(){
528.    for(;;){
529.       key = customKeypad.getKey();
530.       switch(key)
531.       {
532.        case '0' ... '9':
533.            display.write(key);
534.            display.display();
535.            num[0] = num[0]*10 + (key - '0');
536.            ans1=0;
537.          break;
538.
539.          case 'A':
540.          num[0]=ans;
541.
542.          case'=':
543.            display.setTextSize(1);
544.            display.setCursor(0,46);
545.            display.drawLine( 0,  44,  128, 44,
    SSD1306_WHITE );
546.            double l1=0;
547.            if(int(num[0])&1){
548.              for(double k=3;;k=k+2){
549.                l1=k*k;
550.                if(l1>num[0]){
551.                  numa=String(num[0],0);
552.                  display.print(numa);
553.                  display.println(" is Prime");
```

```
554.                   display.display();
555.                   break;
556.                 }
557.                 else if(int(num[0])%int(k)==0){
558.                   numa=String(num[0],0);
559.                   display.print(numa);
560.                   display.println(" is Composite");
561.                   numa=String(k,0);
562.                   display.print(numa);
563.                   display.println(" is a factor");
564.                   ans=k;
565.                   display.display();
566.                   break;
567.                 }
568.               }
569.             }
570.             else if(num[0]==2){
571.               numa=String(num[0],0);
572.               display.print(numa);
573.               display.println(" is Prime");
574.               display.display();
575.             }
576.             else{
577.               int k=2;
578.               numa=String(num[0],0);
579.               display.print(numa);
580.               display.println(" is Composite");
581.               numa=String(k,0);
582.               display.print(numa);
583.               display.println(" is a factor");
584.               ans=k;
585.               display.display();
586.             }
587.             return;
588.       }
589.     }
590.   }
```

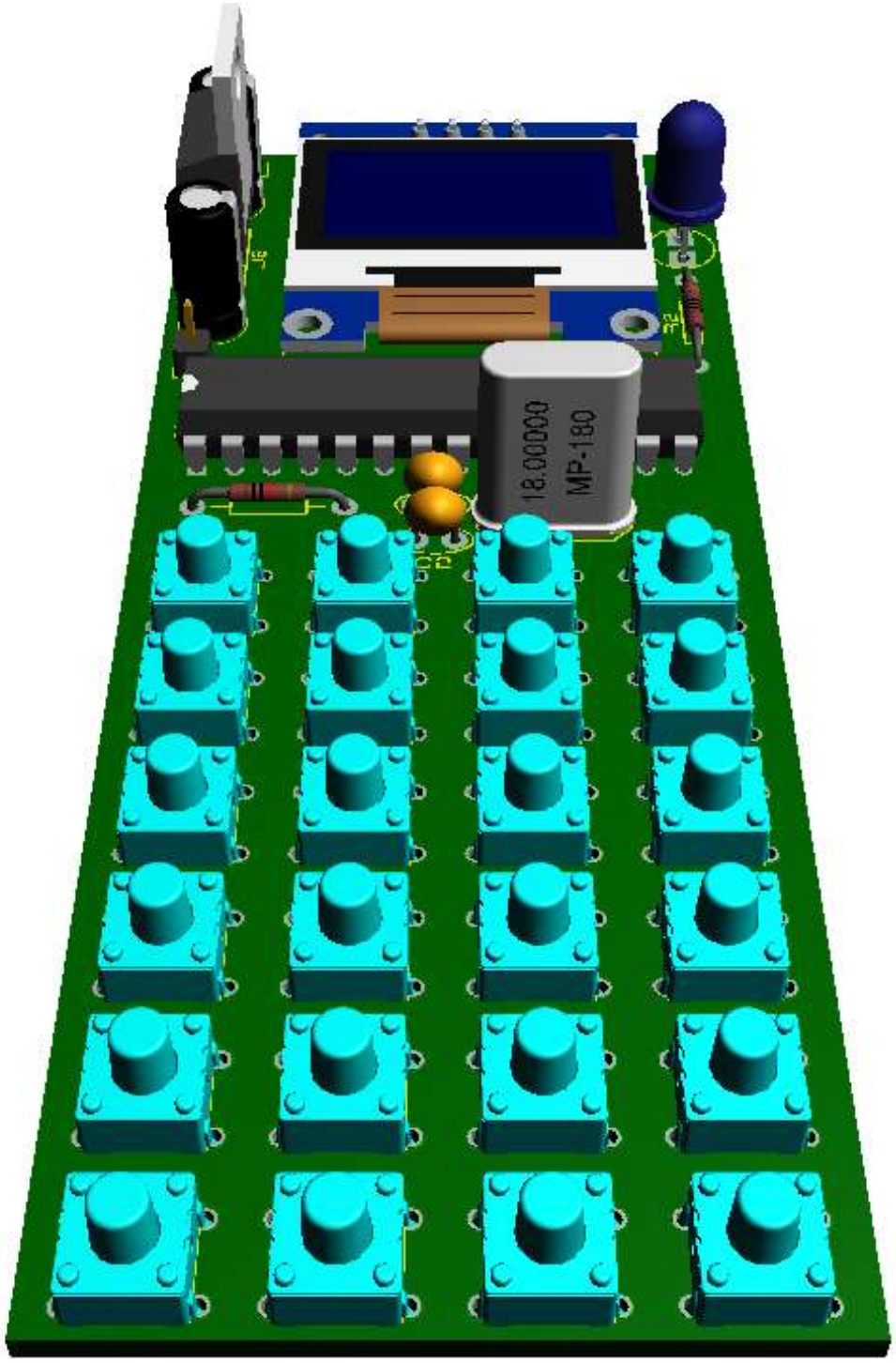
PROCEDURE

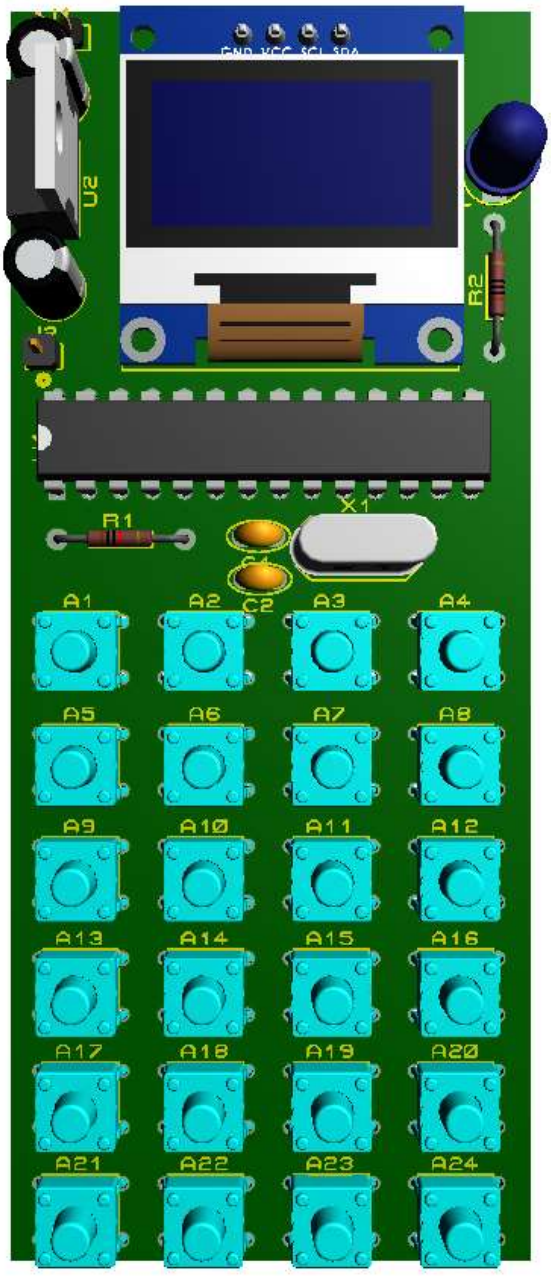
1. Install the Proteus design suite.
2. Instal the Arduino IDE.
3. Type the code you want to upload in the Arduino IDE.
4. Now go to Sketch -> Export Compiled Binary, it will generate the required HEX file.
5. Now open Proteus and make new project.
6. Go to pick components.
7. Select the required components.
8. Select resistor, capacitor, ATmega328p, connectors, button, LED, LCD, IC7805 and other if required.
9. Place the components on the screen.
10. Edit the values of the components as per the requirements.
11. Connect the components according to the schematic diagram given above.
12. Now open the properties of the MCU unit, in program click the browse icon and the select the HEX file without bootloader generated in step 4.
13. Now run the simulation.
14. To make the PCB layout, go to PCB layout.
15. Make PCB by making drawing a 2D rectangle in “Board edge” category.
16. Go to Tools -> Auto-placer.
17. It will automatically place all the elements on the screen.
18. Go to Tools -> Auto-router.
19. It will automatically route all the elements on the screen.
20. If you want custom placement, then edit the component and route.
21. For that, switch to track mode and select the required width of track and start drawing the tracks.
22. For 3D view, go to 3D view and view the 3D view

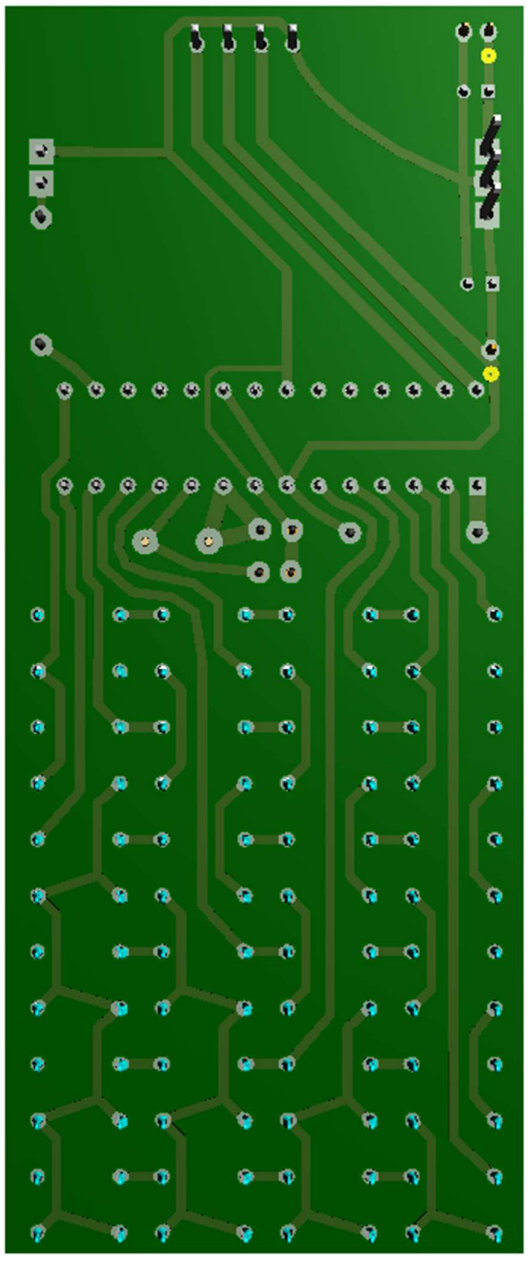
PCB LAYOUT


J1
J3
U2
C4
J2
U1
D2
R2
P1
X1
C1
C2
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
A24

3D LAYOUT

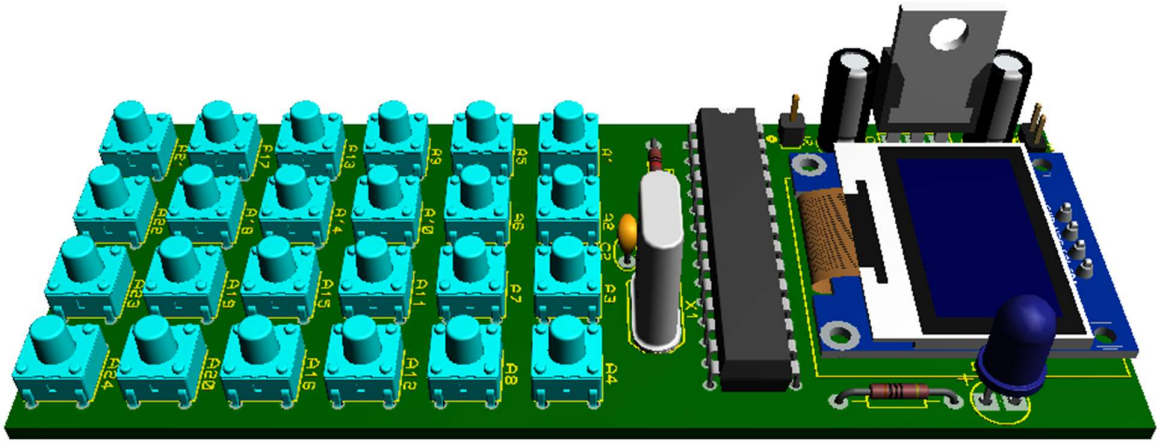




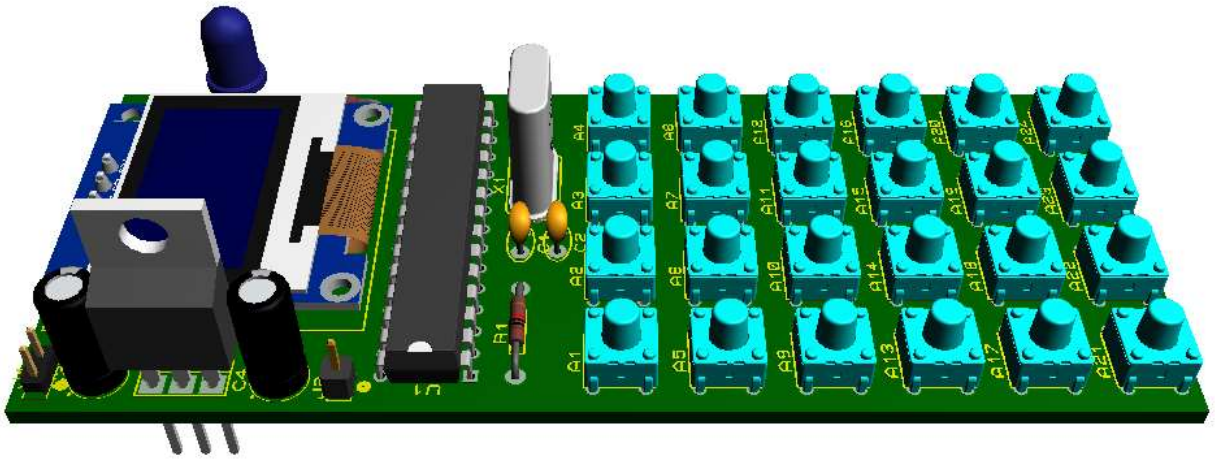


FRONT VIEW BACK VIEW

RIGHT VIEW

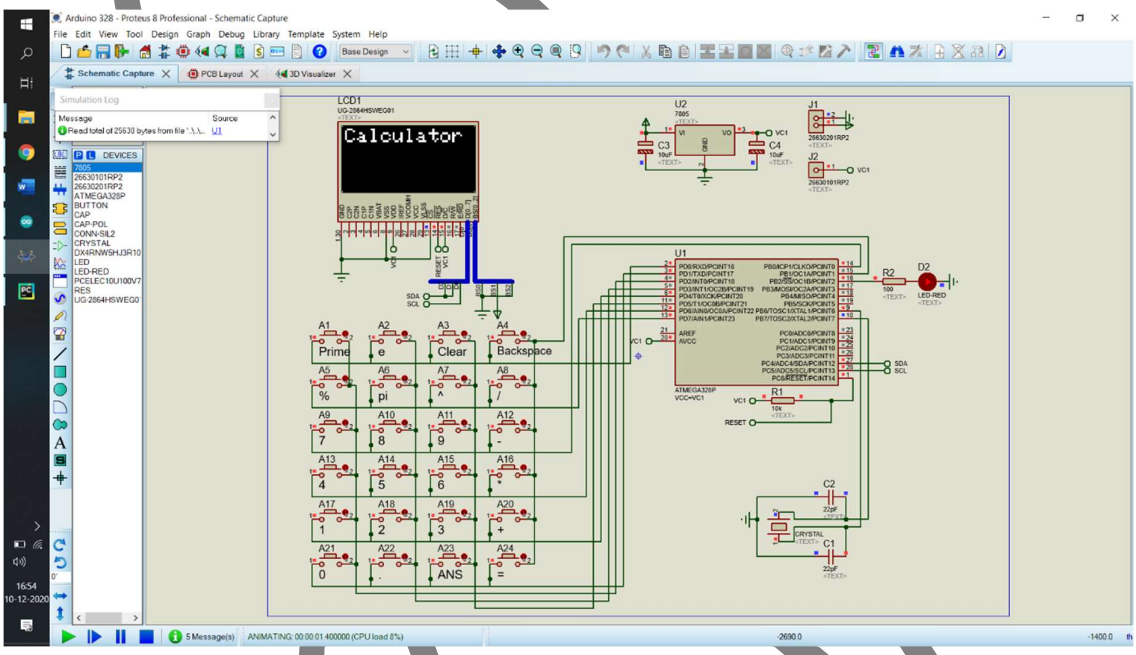


LEFT VIEW

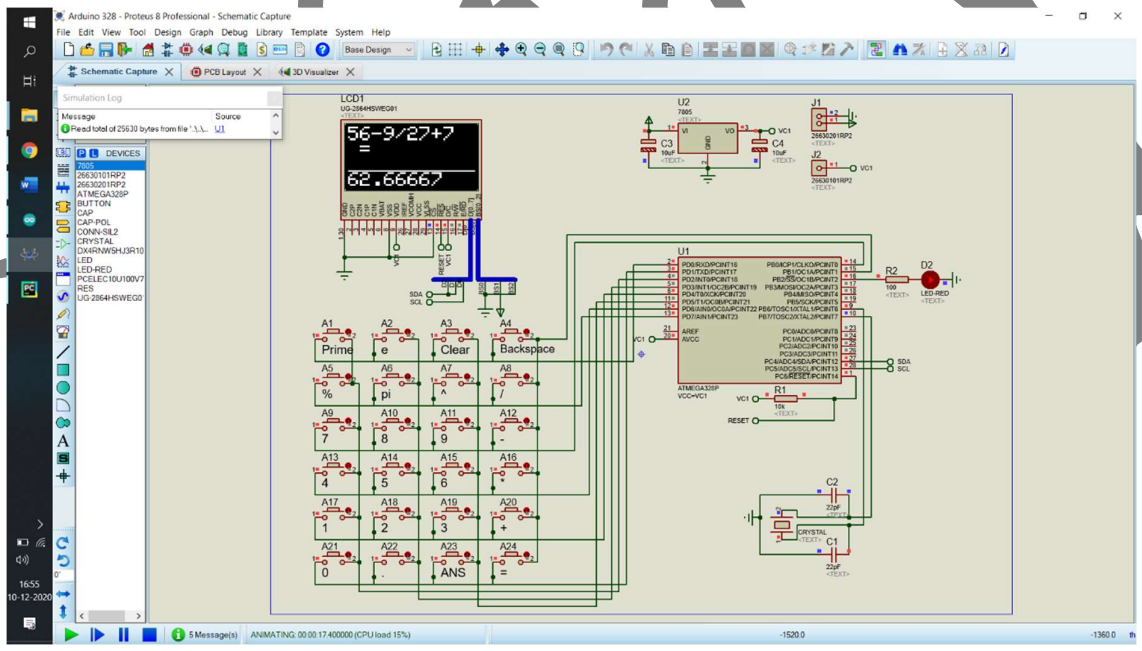


REAL OUTCOME

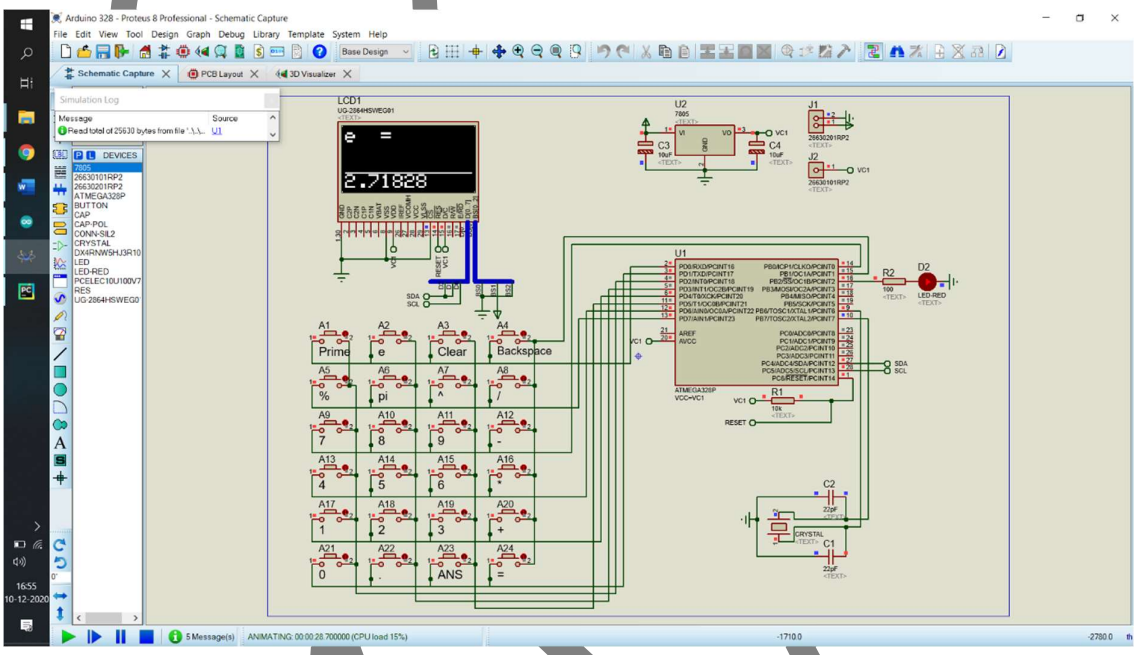
Boot Up screen



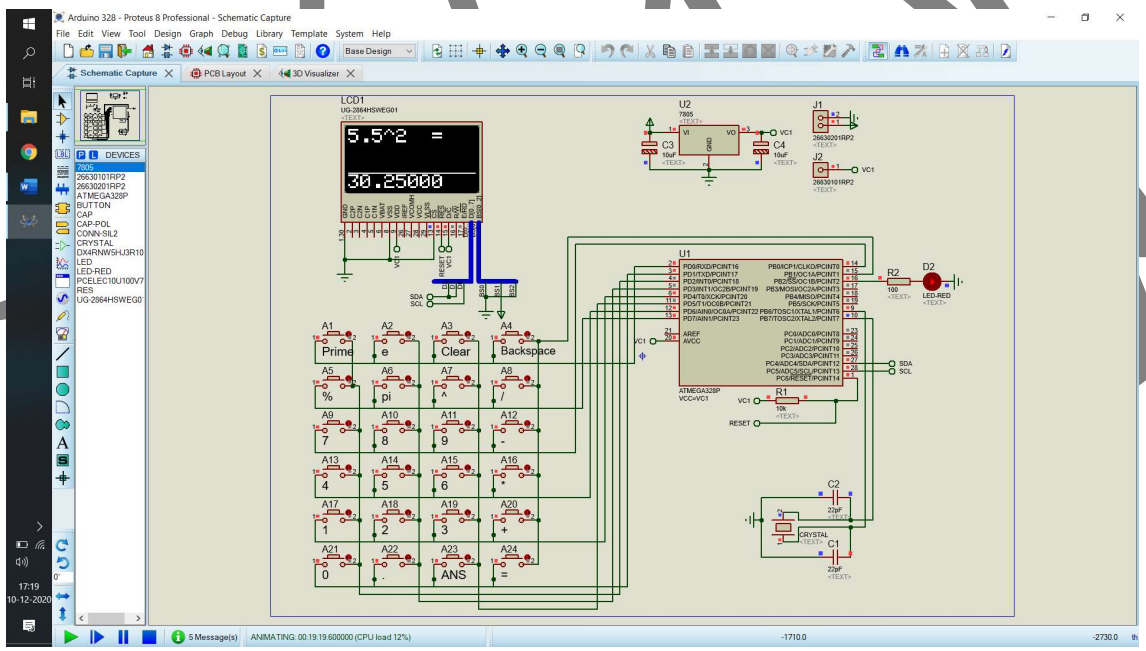
Some calculation



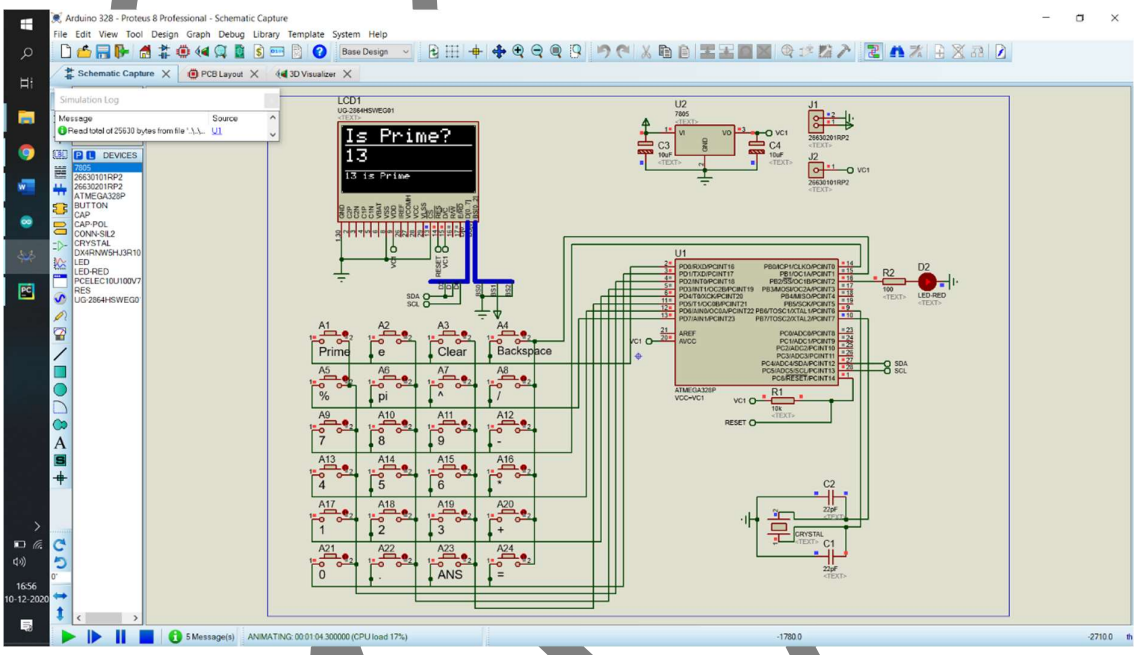
Using constant



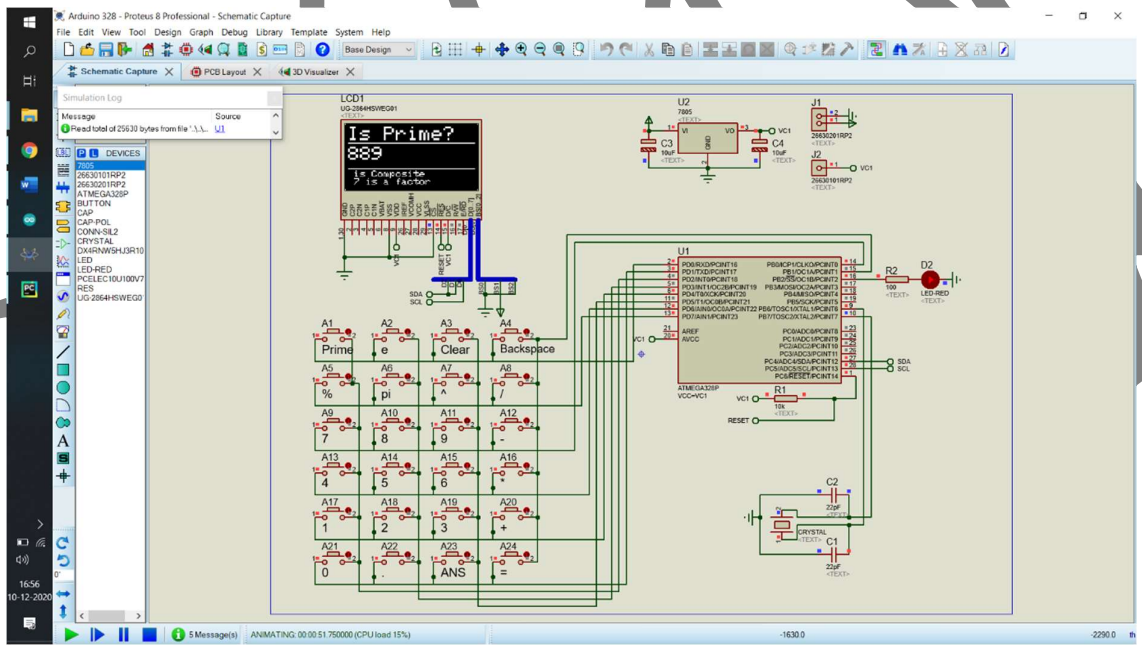
Calculating square of 5.5



Prime Check



Prime Check (since number is composite, factor is returned)



CONCLUSION

Calculator has become an integral part of our daily usage, it's hard to imagine a life without calculator, even if we don't use it directly, it's a foundation for many other products and services. So, we are using calculator directly or indirectly every day.

Here we successfully implemented a calculator using a general microcontroller unit, which is capable of doing many things and is cheap compared to many other options available right now in the market and also it provides many other new functions which are not generally available in options available in market and we have to implement them in 2-3 steps which consumes time and also increases the chances of error. Also, our product has reusability, we can remove the MCU unit and either use if for another purpose or can update the software of the calculator making it upgradable and also if it got damaged we can change it. We can also remove the OLED screen in the physical model, so we can change the screen if any of the pixel got burned and also use it for other purpose if calculator is not in use, and rest of the components are very cheap, other than OLED and MCU, the cost of the components will be around ₹130, which is very cheap compared to other options on the basis of number of functions, reusability, upgradability, repairability and customizability. Its physical size is also very small compared to the current market available options. It has very less weight.

We can also make also our own function in this calculator, like to some buttons to calculate GST, which when pressed will tell the amount after adding GST, every GST percentage can have its own button. By this speed of calculation will increase and also reducing the chances of error. We can also add our own constants, like some fixed cost, which need to be added in any bill irrespective of the quantity purchased. Thus, using the customizability property of our product, which is currently not available in market in reasonable price. We can also print a nice 3D case, if needed.

FUTURE ASPECTS

We can make many amends and upgrade the components of project like

- We can upgrade the MCU unit to a better/faster MCU, which has long range for integers and floats.
- We can also use the QFP package of MCU and SMD package for resistance and capacitors, thus reducing the area occupied by the components and in turn decreasing the size of overall project
- We can use a microprocessor instead of MCU, to increase the speed and memory constraints.
- We can make a switch button, like “shift” in scientific calculators, thus increasing the number of functions implemented
- We can also increase the size of the screen
- We can also add trigonometric functions
- We can also add logarithmic functions
- We can also add different notations to show answer
- We can also add brackets.
- We can also introduce different modes which can be used to calculate the roots of variable equation etc.
- We can also plot the graphs of equations
- We can increase the clock speed of the MCU, to provide fast result
- We can make customized functions which will be having formulas which are used by us in daily life, like we can have simple interest and compound interest formulae entered, after which we will be only required to enter the value as directed and the result will come, we can also add formulae for mean, median etc.