Electronic Writing Pad Using Arduino Uno

Dissertation submitted to

SGTB Khalsa College, Delhi University

For the award of degree of

Bachelor of Science

In

ELECTRONIC SCIENCE

BY

Sanskriti Grover

(2019ELE1035)

Under the supervision of

Dr.P. Arun

Department of Electronics

Sri Guru Tegh Bahadur Khalsa College

University of Delhi

CERTIFICATE

This is to certify that the dissertation entitled “Electronic Writing Pad Using Arduino Uno “ being submitted by Ms. Sanskriti Grover (Exam Roll No. 19068558014) to the SGTB Khalsa College, University of Delhi, Delhi for the award of degree of Bachelor of Science in Electronic Science, is the original work carried out by her. The result in the thesis have not been submitted anywhere in part or full for the award of any other diploma or degree.

Sanskriti Grover
(Candidate)

Dr.P. Arun

Department of Electronics

Sri Guru Tegh Bahadur Khalsa College

University of Delhi, North Campus

New Delhi -110007, India

CONTENTS

Chapter 1: Introduction and Background

1.1 Introduction

Digital technologies are advancing more rapidly in recent years. During the pandemic of 2020, digital education is being seen as an alternative to the traditional education process of chalks and talks. Even assignments and tests were conducted in an online mode, although the writing in notebooks remained traditional. From the early childhood we have been taught to save paper to reduce deforestation but for the same every year so many notebooks and papers are produced. Government of India is also focusing on going digital in order to prevent wastage of paper. A hidden advantage of digitalization is the conservation of environment. If these notebooks are replaced by the digital writing pads, then we would be able to reduce the paper production to a greater extent.

There are various tablets in market which provide writing options, however there is no tablet custom designed for students who would use it for longer hours. Thus, there is a need for dedicated electronic gadget designed for this purpose. The design of this electronic gadget should be portable and accessible according to the comfort of the students. As using the screen for longer period of time put strain on eyes, so the design of the gadget should focus on use of appropriate wavelength of color which does not harm the eye sight of the user. The device should use the colors of larger wavelength, because lights of shorter wavelengths are hard on eyes since they produce more energy [1].

While designing portable devices, size to power ratio is always a concern. Designing the tablet with commonly available microcontroller helps to make it portable and efficient yet keeping the cost manageable. The design should allow the microcontroller to last for a longer period of time on one charge, which will allow it to be kept in use for long period of time without any form of maintenance. Along with the microcontroller the device should have a local storage which would allow the user to save their data.

1.2 Background

We intend to implement this project with Arduino Uno considering its easy availability, easy configurability and extensive documentation. It's an inexpensive microcontroller, with a simple Integrated Development Environment and easy programmable options [2]. It is easy-to-use open-source microcontroller board which can be programmed using the Arduino Integrated Development Environment (IDE). Arduino Uno is capable of handling display units very well, which ensures easy communication between Arduino and display.

For the display, the Thin Film Transistor (TFT) technology with LCD is used to improve the image quality. In this technology, Transistors are embedded within the panel itself, which reduces the crosstalk between pixels and improve image stability [3]. The TFT LCD modules are compatible with Arduino Uno and Arduino Mega. The pins of this shield are designed to be easily installed on the Arduino Uno boards. Thus, these modules do not require any wiring for the connections with Arduino Uno.

The data on the display must be stored for future. Therefore, for protecting user's data for a longer period of time a storage device must be used. Micro SD card is the memory card which can be used along with the TFT display module to store user's data in readable format. This will enable user to save their data for future reference.

The design of the device should ensure that it does not require constant maintenance in the form of using the USB port or external power source for Arduino. The chargeable batteries can be used to avoid plugging the device repeatedly for its use. This will ensure longer durability and allow the device to last for a longer period of time on one charge.

This project is aimed at migrating the education sector to digital platform to reduce the paper production in order to keep the environment healthy.

CHAPTER 2: Components Description

2.1 Arduino

Arduino is an open-source electronics platform, based on easy-to-use hardware and software. Arduino provides both a physical programmable circuit board and a programming environment, particularly known as Arduino IDE (Integrated Development Environment). Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or Breadboards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

2.1.1 Arduino Uno

Arduino Uno is a microcontroller board based on ATmega328P. It can be powered via the USB connection or with an external power supply. This board comes with a built-in regulation feature which keeps the voltage under control when the device is connected to the external device. It has 20 input/output pins, out of which 14 are digital and 6 are analog. Out of 14 digital pins 6 pins can be used for PWM outputs. There is also a reset pin on the board which reset the whole board. The Arduino boards only required 5V to turn the board ON which can be achieved by USB port or external adapter or external power source [4].

2.1.2 Components of Arduino Uno board

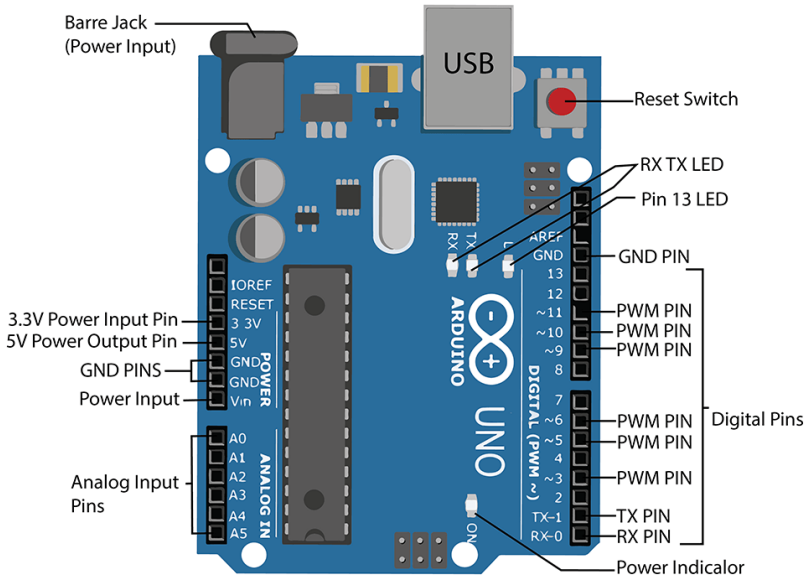


Fig 1. Pin description of Arduino Uno

The components of Arduino Uno board are as follows:

- **Power USB** - Arduino board can be powered by using the USB cable from computer. It is also used for loading programs from personal computers.
- **Voltage Regulator** - The voltage regulator controls the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
- **Power LED indicator** - This LED lights up when Arduino is plugged into a power source, it indicates that board is connected correctly.
- **Reset Switch**- Arduino board can be reset in two ways. First, by using the reset switch available on board. Second, by connecting an external reset button to the Arduino pin labelled 'RESET'.
- **TX and RX Pins** - Serial: 0 (RX) and 1 (TX) are used to receive (RX) and transmit (TX) TTL serial data.
- **External Power Supply**- External power of 9 –12V can be given to Arduino board .

Digital pins

The Arduino UNO board has 14 digital I/O pins out of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled "~" can be used to generate PWM.

Analog pins

The Arduino UNO board has six analog input pins from A0 to A5. These pins read signals from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

Power Pins

- VIN: This is used to power the Arduino board from an external power source, like AC mains power supply.
- 5V: This the regulated power supply used to power other components on the board. This can be supplied either from an on-board regulator, or from USB.
- 3.3V: A 3.3-volt supply generated by the on-board FTDI chip.
- GND: Ground pins.

2.1.3 Memory features

All the different memory units inside a microcontroller can be divided into two main types: **RAM** and **ROM**. RAM (Random-Access Memory) in microcontroller-based systems is a volatile memory used to store temporary data such as the system's firmware variables. ROM (Read-Only Memory) in microcontroller-based systems is non-volatile memory used to store permanent data such as the system's firmware.

RAM and ROM in microcontroller-based systems are organized into three main categories:

- Flash memory: It is the memory which stores the system's firmware which needs to be executed.
- SRAM (static random access memory): SRAM is where the sketch creates and manipulates variables when it runs.
- EEPROM: It is memory space that programmers can use to store long-term information.

Flash memory and EEPROM memory are non-volatile. SRAM is volatile and will be lost when the power is disconnected.

The memory features of Arduino Uno are:

- Flash memory - Arduino Uno has flash memory of 32kb. The bootloader takes 2kb flash memory, so it has only 30kb of flash memory.
- SRAM - Arduino Uno has SRAM of 2kb
- EEPROM memory – Uno has EEPROM of 1Kb [5].

2.2 TFT LCD

A thin film transistor liquid crystal display is a type of color display which makes use of thin-film transistor technology in order to improve qualities such as contrast and addressability of the display. TFT technology means that an individual transistor is used to drive each individual pixel, allowing faster response time. This technology can be seen in tablets, TVs, and countless other devices.

The TFT LCD controls individual pixels in the display by setting the level of the electric field across the three liquid crystal capacitors (one for each sub-pixel of red, green and blue) in the pixel in order to control the polarization of the crystal material. The amount of polarization in the crystal determines the amount of light that reaches the color filter from the backlight. The display screen is composed of many pixels that can emit light of any color, and the purpose can be achieved by controlling each pixel to display the corresponding color.

A TFT LCD has a liquid crystal layer between a glass substrate formed with TFTs and transparent pixel electrodes and another glass substrate with a color filter (RGB) and transparent

counter electrodes. In addition, polarizers are placed on the outer side of each glass substrate and a backlight source on the back side. A change in voltage applied to liquid crystals changes the transmittance of the panel including the two polarizing plates, and thus changes the quantity of light that passes from the backlight to the front surface of the display. This principle allows the TFT LCD to produce full-color images [7].

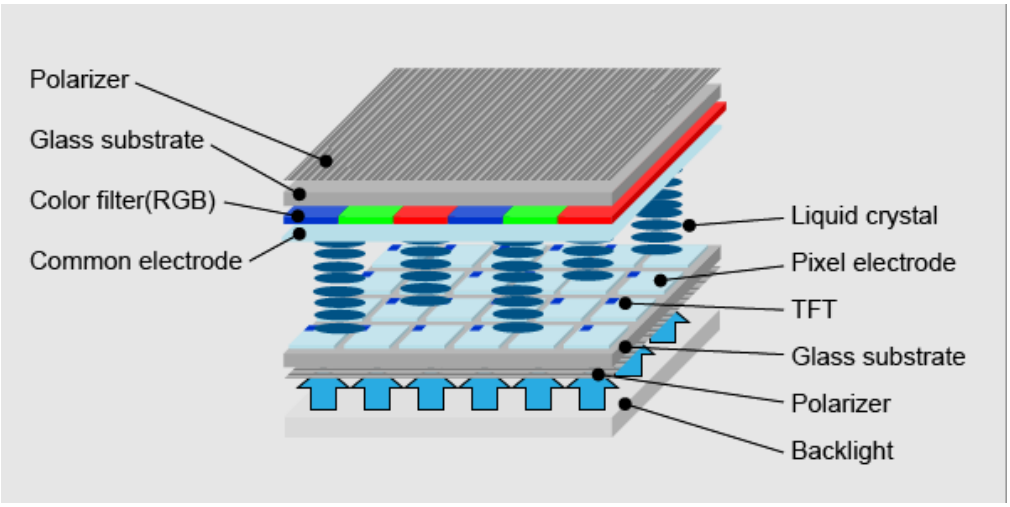


Fig 2: (a) Structure of TFT LCD.

Fig 2(b) 3.5 inches TFT LCD.

2.2.2 ILI9341 TFT LCD

The ILI9341 is a 3.5inch, 480*320 resolution display, which supports capacitive touch control. It operates on 5V/3.3V and parallel bit module which is faster than Serial Peripheral Interference (SPI). ILI9341 supports 8-color display mode and sleep mode for precise power control by software and these features make the ILI9341 an ideal LCD for medium or small size portable devices such as digital cellular phones, smart phone. This TFT LCD display can be directly attached to Arduino Uno or Arduino Mega. [6]

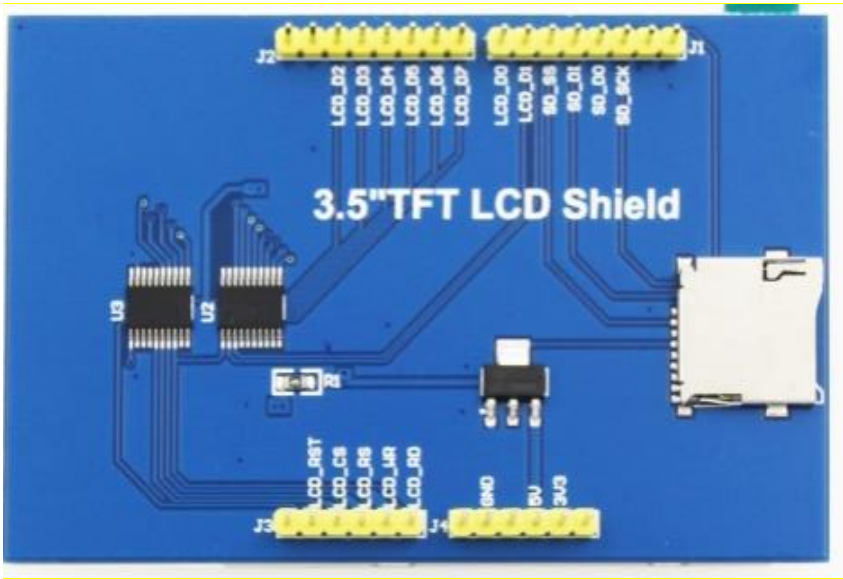


Fig 3 Pin description of TFT LCD

Pin description of TFT LCD -

1	LCD_RST	LCD bus reset signal, low level reset
2	LCD_CS	LCD bus chip select signal, low level enable
3	LCD_RS	LCD bus command / data selection signal, low level: command, high level: data
4	LCD_WR	LCD bus write signal
5	LCD_RD	LCD bus read signal
6	GND	Power ground
7	5V	5V power input
8	3V3	3.3V power input, this pin can be disconnected
9	LCD_D0	LCD 8-bit data Bit0
10	LCD_D1	LCD 8-bit data Bit1
11	LCD_D2	LCD 8-bit data Bit2
12	LCD_D3	LCD 8-bit data Bit3
13	LCD_D4	LCD 8-bit data Bit4
14	LCD_D5	LCD 8-bit data Bit5
15	LCD_D6	LCD 8-bit data Bit6
16	LCD_D7	LCD 8-bit data Bit7
17	SD_SS	SD card SPI bus chip select signal, low level enable
18	SD_DI	SD card SPI bus MOSI signal
19	SD_DO	SD card SPI bus MISO signal
20	SD_SCK	SD card SPI bus clock signal

2.3 SD Card

SD card stands for Secure Digital Memory Card, it is a tiny flash memory card designed for high-capacity memory for various portable devices, cellular phones, digital cameras smartphones. An SD card features a high data transfer rate and low battery consumption, which is the primary considerations for portable devices. An SD card uses flash memory to provide nonvolatile storage, which means a power source is not required to retain stored data. Also, these cards use metal connector contacts, instead of the traditional pins and plugs, so they aren't as prone to damage during handling.

Chapter 3: Implementation

3.1 Hardware Implementation

The hardware requirements include interfacing of TFT LCD and SD card with Arduino Uno. The Microcontroller is responsible for providing commands to TFT LCD and collecting user's input. The SD card module's role is to provide memory space to store user's data displaying on TFT LCD for long periods of time.

3.1.1 Interfacing of Arduino with ILI9341 TFT LCD

The 3.5-inch display is a ready-made shield for Arduino Uno, which can also be placed on the Arduino Mega. The pins of this shield are designed to be easily installed on the Arduino Uno or Arduino Mega.

The ILI9341 TFT display controller operates on 3.3V only. The display module is supplied with 5V that comes from the Arduino board, but this module has a built-in 3.3V regulator which only allows 3.3V from the 5V source to the TFT display module. [8]

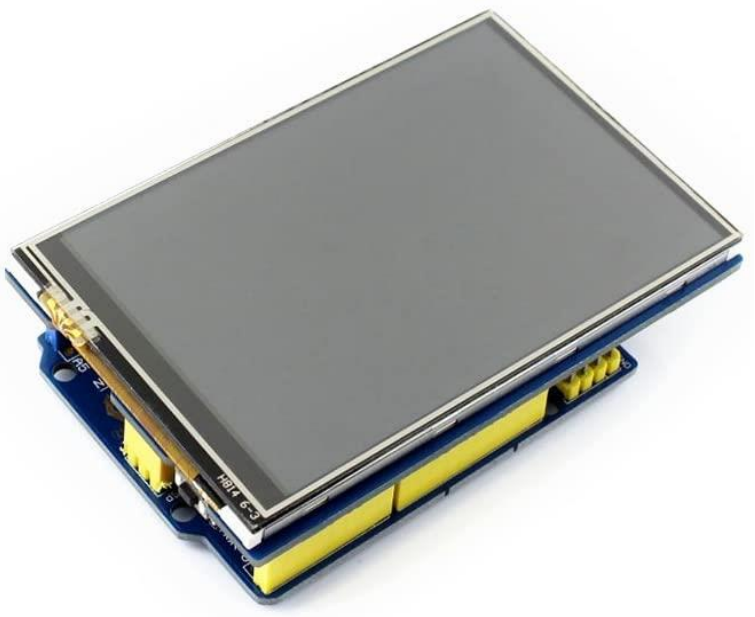


Fig 4(a): 3.5 inches TFT LCD attached to Arduino UNO board

The ILI9341 TFT display has 14 pins, out of which 9 pins are for display and other 5 pins are for touch module. So, the display side pins which numbered from 1 to 9(from left to right): are VCC (5V), GND (ground), CS (chip select), RST (reset), DC (or D/C: data/command), MOSI (or SDI), SCK (clock), BL (back light LED) and MISO (or SDO) [9].

Pin connections of the TFT LCD with Arduino Uno:

TFT LCD Pins	Arduino Pins
LCD_RST	Analog Pin A4
LCD_CS	Analog Pin A3.
LCD_RS	Analog Pin A2
LCD_WR	Analog Pin A1
LCD_RD	Analog Pin A0
GND	GND
5V and 3.3 V	5V and 3.3 V
LCD_D0	Digital Pin 8
LCD_D1	Digital Pin 9
LCD_D2	Digital Pin 2
LCD_D3	Digital Pin 3
LCD_D4	Digital Pin 4
LCD_D5	Digital Pin 5
LCD_D6	Digital Pin 6
LCD_D7	Digital Pin 7

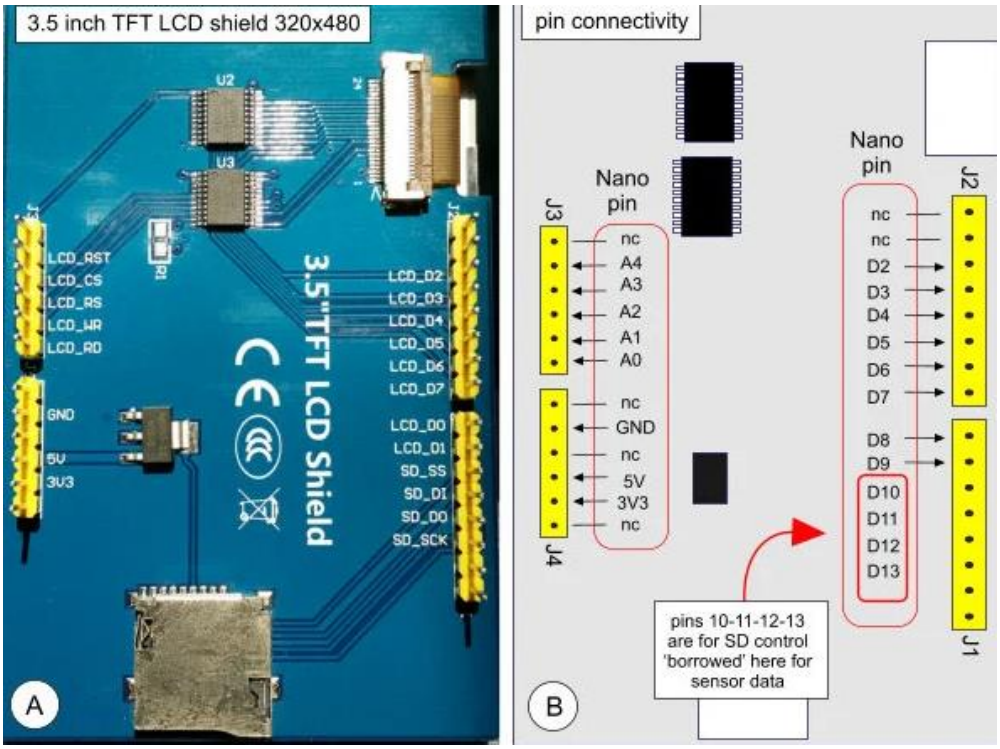


Fig 4(b): Connections

3.1.2 Interfacing of SD card with Arduino

To store data in SD card we should SD card module which allow us to communicate with the memory card and allows to perform read/write operations. As we have seen above that, the TFT LCD display already has SD card slot, so there is no need to use SD card module explicitly.

The SD card interfaces in the Serial peripheral interface (SPI) protocol. The SPI protocol runs using a master/slave set-up and it can run in a full duplex mode (i.e. signal can be transmitted using master and slave simultaneously). [10]

The Pins of ILI9341 TFT LCD for SD card connections with Arduino Uno:

TFT LCD Pins	Arduino Pins
SD_SS	Digital Pin 10
SD_D1	Digital Pin 11
SD_D0	Digital Pin 12
SD_SCK	Digital Pin 13

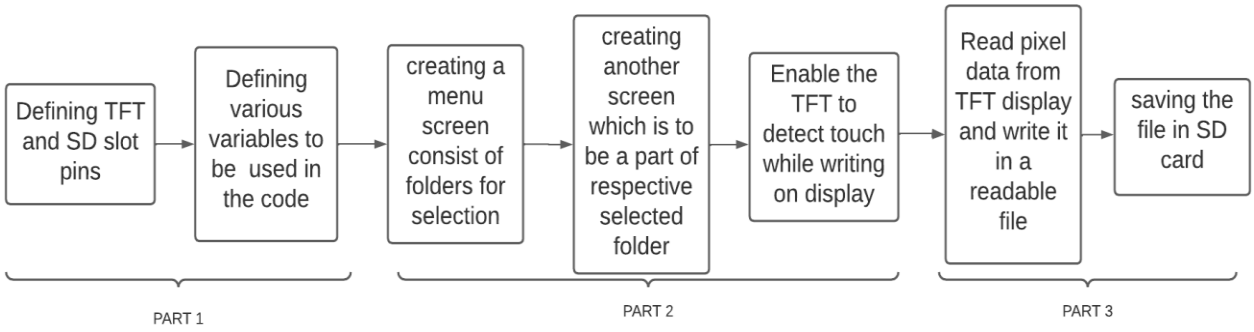
3.2 Software Implementation

The software design of the writing tablet is divided into three parts:

PART 1- Defining the pins and constant.

PART 2 - Creating area for writing.

PART 3 - Reading and saving TFT display data.



3.2.1 Library Selection

For TFT LCD 'Adafruit_GFX' library is used considering its simplicity, easy to use syntax and set of graphics functions for all of TFT LCD, OLED displays and LED matrices [11].

For TFT LCD the following three libraries are used in the source code of Arduino:

- 'Adafruit_GFX' library - core graphic library.
- 'Adafruit_TFTLCD' library - hardware specific library
- 'TouchScreen' library

For SD card, the 'SD library' and 'SPI library' is used.

3.2.2 Architecture of Arduino Program

- **Defining pins and variables –**

First we declare the pins and import the necessary libraries. After that we need to define constants and variables which are to be used throughout the program.

- **Initialization –**

  First we initialize the SD card and TFT LCD screen and then we design the initial screen of TFT display which appears when we start the display. The program of initial screen of the display, which appears on start should be kept inside the setup () function of the Arduino sketch. Arduino Microcontrollers requires setup () function to be defined in its code. In this function the code runs only once. So those parts of the program are kept here which needs to be run only once.

- **Creating menu and main screen –**

  The menu screen will have tabbed items which will be redirected to the main writing area. These lines of code will be kept in a loop to provide repeated access to the functionality.

- **Reading pixel data -**

  For saving the TFT data in SD card in readable format, Arduino should be able to read pixel data of TFT display. But the 'ADAFRUIT_TFTLCD' library has no function for driver ILI9341 to read pixel data. However, it provides function for other TFT LCD driver including 0x9325 and for 0x7575 but for the driver 0x9341(or ILI9341) the code for 'read pixel' is missing in 'ADAFRUIT_TFTLCD' in the cpp file.

NOTE: The 'ADAFRUIT_TFTLCD' library is incomplete or have some missing code for some drivers. As It does not have the 'readpixel' code for the driver ILI9341 so we have to add the code manually in order to get the TFT pixel data in Arduino source code.

Here is the picture of 'READ PIXEL' code in cpp file of 'ADAFRUIT_TFTLCD' library:

```
uint16_t Adafruit_TFTLCD::readPixel(int16_t x, int16_t y) {

  if((x < 0) || (y < 0) || (x >= _width) || (y >= _height))
return 0;

  CS_ACTIVE;
  if(driver == ID_932X) {


    uint8_t hi, lo;
    int16_t t;
    switch(rotation) {
     case 1:
      t = x;
      x = TFTWIDTH  - 1 - y;
      y = t;
      break;
     case 2:
      x = TFTWIDTH  - 1 - x;
      y = TFTHEIGHT - 1 - y;
      break;
     case 3:
      t = x;
      x = y;
      y = TFTHEIGHT - 1 - t;
      break;
    }
    writeRegister16(0x0020, x);
    writeRegister16(0x0021, y);
    // Inexplicable thing: sometimes pixel read has high/low
bytes
    // reversed.  A second read fixes this.  Unsure of reason.
Have

      // tried adjusting timing in read8() etc. to no avail.
      for(uint8_t pass=0; pass<2; pass++) {
        CD_COMMAND; write8(0x00); write8(0x22); // Read data
   from GRAM
        CD_DATA;
        setReadDir();  // Set up LCD data port(s) for READ
   operations
        read8(hi);     // First 2 bytes back are a dummy read
        read8(hi);
        read8(hi);     // Bytes 3, 4 are actual pixel value
        read8(lo);
        setWriteDir(); // Restore LCD data port(s) to WRITE
```

```
configuration
    }
    CS_IDLE;
    return ((uint16_t)hi << 8) | lo;

  } else if(driver == ID_7575) {

    uint8_t r, g, b;
    writeRegisterPair(HX8347G_COLADDRSTART_HI,
HX8347G_COLADDRSTART_LO, x);
    writeRegisterPair(HX8347G_ROWADDRSTART_HI,
HX8347G_ROWADDRSTART_LO, y);
    CD_COMMAND; write8(0x22); // Read data from GRAM
    setReadDir();  // Set up LCD data port(s) for READ
operations
    CD_DATA;
    read8(r);      // First byte back is a dummy read
      read8(r);
      read8(g);
      read8(b);
      setWriteDir(); // Restore LCD data port(s) to WRITE
 configuration
      CS_IDLE;
      return (((uint16_t)r & B11111000) << 8) |
             (((uint16_t)g & B11111100) << 3) |
             (           b              >> 3);
   }
```

The above images shows that the 'readpixel' function in cpp file of 'ADAFRUIT_TFT' LIBRARY does not have code for the driver ILI9341. So first the code for 'readpixel' for ILI9341 driver should be written in cpp file of the 'Adafruit_TFTLCD' library.

This code can be extracted from the modified version of 'ADAFRUIT_TFTLCD' library from Arduino's official website.

The code of cpp file for the driver ILI9341 is given below:

```
else if(driver == ID_9341) {
  uint8_t r, g, b;
#define ILI9341_MEMORYREAD      0x2E
  setAddrWindow( x,y,x,y);
  CS_ACTIVE;
  CD_COMMAND;
  write8(ILI9341_MEMORYREAD);
  setReadDir();  // Set up LCD data port(s) for READ operations
  CD_DATA;
  read8(r);     // First byte back is a dummy read
  read8(r);
```

```
  read8(g);
  read8(b);
  setWriteDir(); // Restore LCD data port(s) to WRITE configuration
  CS_IDLE;
  return (((uint16_t)r & B11111000) << 8) |
      (((uint16_t)g & B11111100) << 3) |
      (       b          >> 3);
}
else
return 0;
}
```

- **Saving pixel data in SD card**

  In order to save the collected pixel data, from TFT display, we should create a file in a readable image format. So the file should be in BMP, JPG or PNG format.

  The JPG and PNG require computational power for decoding. Also these files are compressed and lossless. On the other hand, BMP files are uncompressed and can handle as much detail as possible. Therefore, we prefer BMP format because they have higher quality and well documented than JPG and PNG [12].

The BMP file format consists of:

Name	Size	Description
Header	14 bytes	It contains information about the type, size, and layout of a device-independent bitmap file
Info – Header	40 bytes	It specifies the dimensions, compression type, and color format for the bitmap
Color table	12 bytes	To get the color of the pixel . Each pixel bit represents the Red- Green –Blue (RGB) values in the actual bitmap data area
Pixel data	width * height * 2 bytes	It represents the image data

So, the total file size of the BMP image so formed can be calculated as:

File size = 14+40+12+w*h*2

where w and h represents the width and height of the TFT [13].

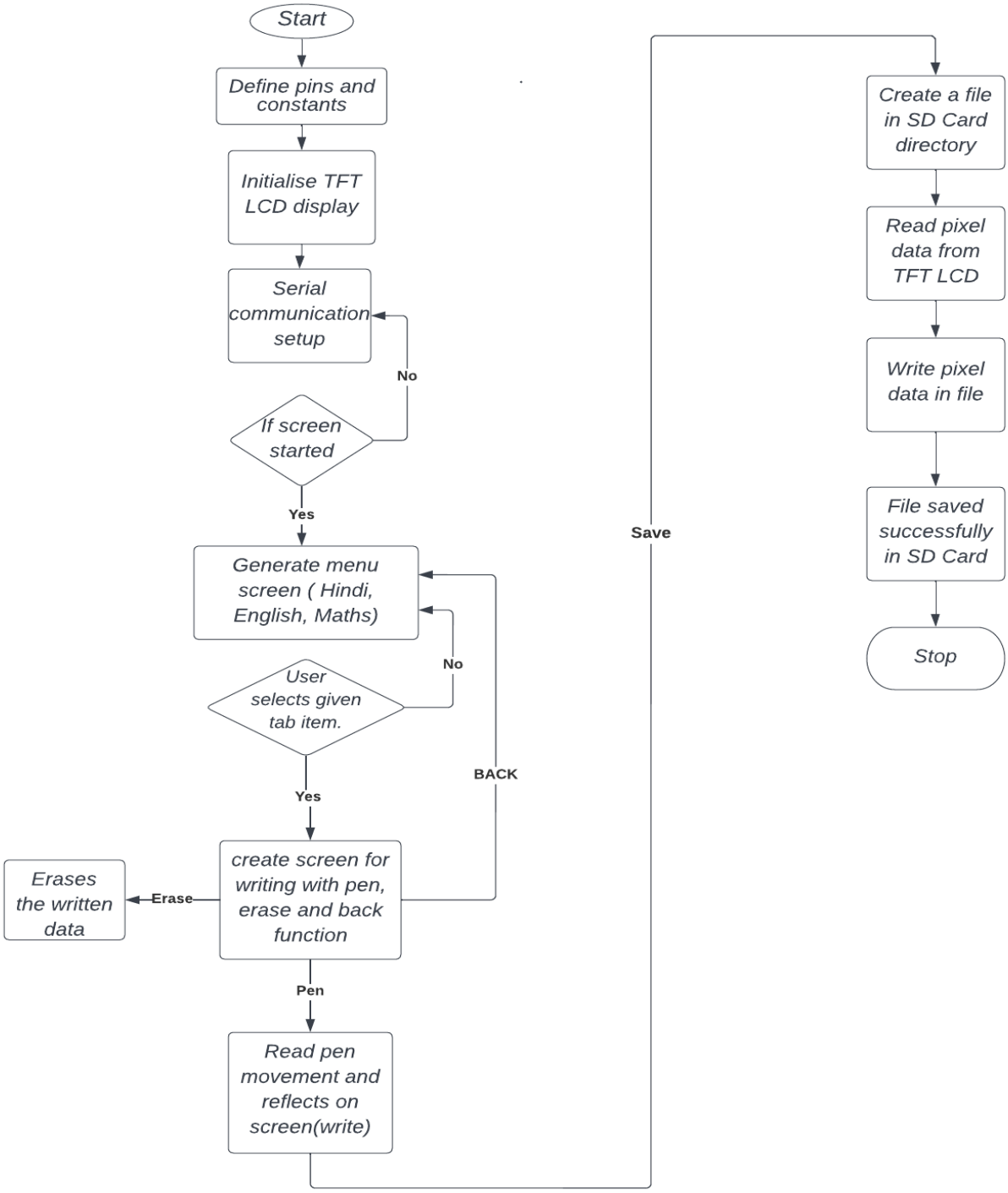
3.3 Serial communication

To integrate the hardware and software, serial communication is used. It is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. For every touch on the TFT LCD display, a particular command is sent to the microcontroller which perceives the touch on the basis of coordinates (x, y) and responds according to the input send at the display.[14]

Chapter 4: Programming

4.1 Arduino Programming

4.1.1 Flow Chart

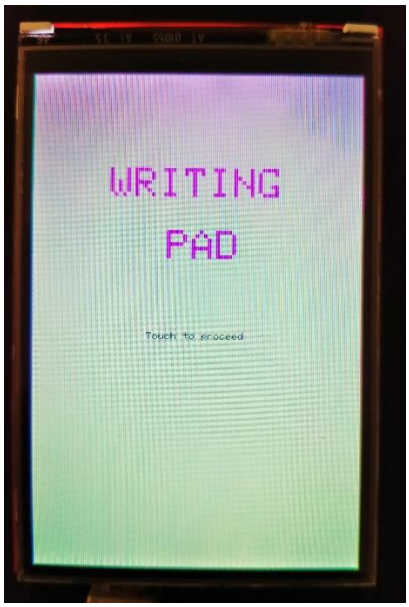


Chapter 5: Results

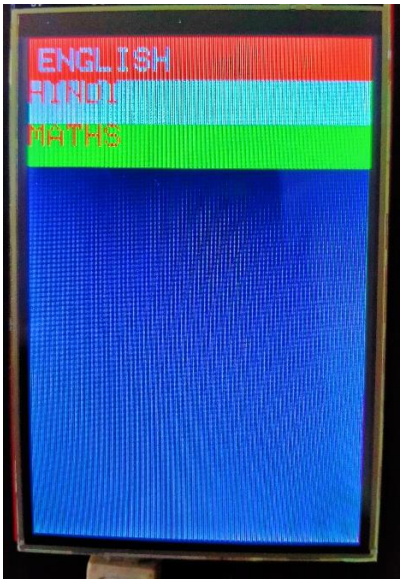
The Electronic Writing Pad using Arduino Uno is Designed successfully and tested for different test cases.

The writing on the screen was successfully tested and here are some results:

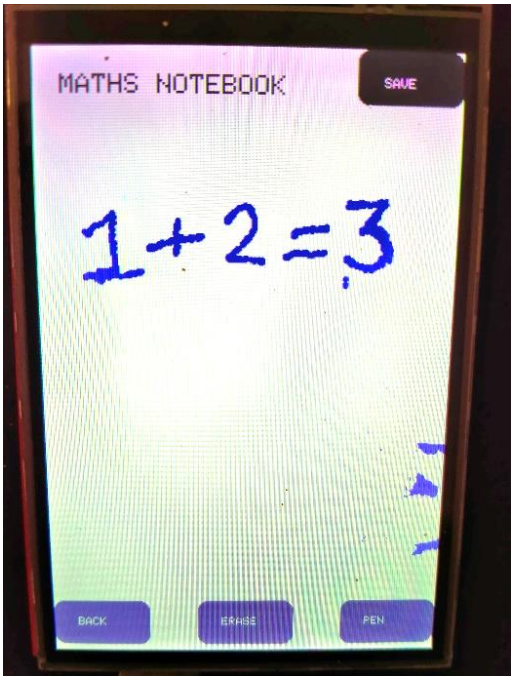
Initial Screen:



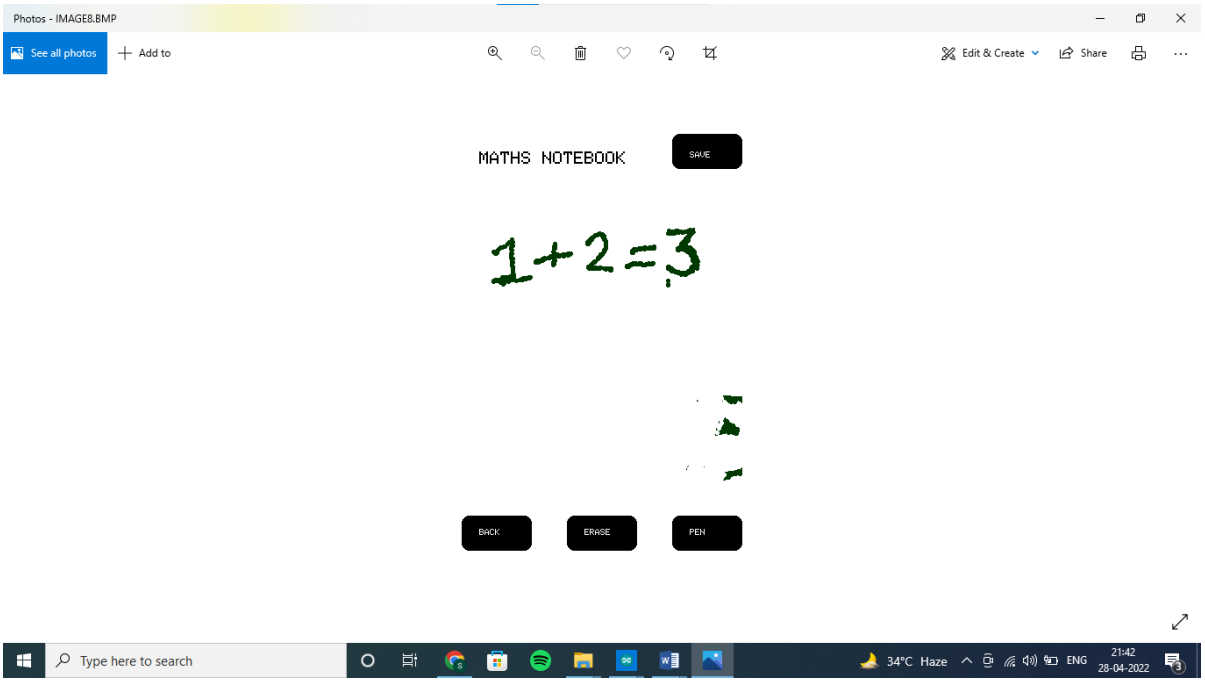
Menu Screen :



Main screen:



Also, the screen was successfully saved in SD card :



File size

The file size is:

Header - 14 bytes

Info header - 40 bytes

Color mask- 12 bytes

Image data - w *h*2

The total file size should be: 14+40+12+w*h*2

The dimension of the TFT display is 320 x 480 but we are reading the pixel data for 320*479

So the file size = 14+40+12+ 320*479*2

= 306,626 bytes

The purpose of different libraries in Arduino IDE and how to make some changes in the library is the important context of this dissertation. While creating file in image format, how the file size can be calculated for different image format has been learnt. Also, got the knowledge of SPI protocols along with its advantages and disadvantages.

APPENDIX 1

Code

```
#include <SPFD5408_Adafruit_GFX.h>    // Core graphics library
#include <SPFD5408_Adafruit_TFTLCD.h> // Hardware-specific library
#include <SPFD5408_TouchScreen.h>
#include <SPI.h>
#include <SD.h>

#if defined(__SAM3X8E__)
#undef __FlashStringHelper::F(string_literal)
#define F(string_literal) string_literal
#endif

#define YP A1 //
#define XM A2
#define YM 7
#define XP 6



// Calibrate values
#define TS_MINX 940
#define TS_MINY 100
#define TS_MAXX 125
#define TS_MAXY 970

//the resistance between X+ and X-, Used multimeter to read it
// For the one we're using, its 300 ohms across the X plate
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
```

```
#define LCD_CS A3
#define LCD_CD A2
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4
#define bmpDraw
//  readable names to some common 16-bit color values:
#define  BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);

#define BOXSIZE 40
#define PENRADIUS 2
#define ERASE  4
#define MINPRESSURE 10
#define MAXPRESSURE 1000
#define SD_CS 53
int oldcolor, currentcolor;
int oldfd, currentfd;
int currentpage;
```

```
//**WAIT ONE TOUCH**//
TSPoint waitOneTouch() {

  TSPoint p;
  do {
    p = ts.getPoint();

    pinMode (XM, OUTPUT);
    pinMode (YP, OUTPUT);

  } while ((p.z < MINPRESSURE) || (p.z > MAXPRESSURE));

  return p;
}



//**MENU SCREEN **//

void screen ()
{
  tft.fillScreen(BLACK);

  tft.fillRect(0, 0, BOXSIZE * 9, BOXSIZE, RED);
  tft.setCursor (10, 10);
  tft.setTextSize (3);
  tft.setTextColor(WHITE);
  tft.println("ENGLISH");
```

```
  tft.fillRect(0, BOXSIZE, BOXSIZE * 9, BOXSIZE, WHITE);
  tft.setCursor (0, BOXSIZE);
  tft.setTextSize (3);
  tft.setTextColor(RED);
  tft.println("HINDI");

  tft.fillRect(0, BOXSIZE * 2, BOXSIZE * 9, BOXSIZE, GREEN);
  tft.setCursor (0, BOXSIZE * 2);
  tft.setTextSize (3);
  tft.setTextColor(RED);
  tft.println("MATHS");
}


//**SUB MENU SCREEN**//

void english ()
{ currentcolor = BLUE;
  tft.drawRoundRect(0, 0, 320, 480, 10, WHITE);
  tft.fillScreen(WHITE);
  tft.setCursor (20, 20);
  tft.setTextSize (2);
  tft.setTextColor(BLACK);
  tft.println("ENGLISH NOTEBOOK");
  back();
  erase();
  pen();
  save();
}
```

```
void hindi ()
{
  tft.drawRoundRect(0, 0, 320, 480, 10, WHITE);
  currentcolor = BLUE;
  tft.fillScreen(WHITE);
  tft.setCursor (20, 20);
  tft.setTextSize (2);
  tft.setTextColor(BLACK);
  tft.println("HINDI NOTEBOOK");
  back();
  erase();
  pen();
  save();
}

void maths()
{
  currentcolor = BLUE;
  tft.drawRect(0, 0, 320, 480, YELLOW);
  tft.fillScreen(WHITE);
  tft.setCursor (20, 20);
  tft.setTextSize (2);
  tft.setTextColor(BLACK);
  tft.println("MATHS NOTEBOOK");
  back();
  erase();
  pen();
  save();
```

```
}

//**REQUIRED BUTTONS ON SCREEN*//

void erase() {
  tft.drawRoundRect(BOXSIZE * 3, 435, BOXSIZE * 2 , BOXSIZE , 8, WHITE);
  tft.fillRoundRect(BOXSIZE * 3, 435, BOXSIZE * 2 , BOXSIZE , 8, BLACK);
  tft.setCursor (140, 450);
  tft.setTextSize (1);
  tft.setTextColor(WHITE);
  tft.println("ERASE ");

}

void pen() {
  tft.drawRoundRect(BOXSIZE * 6, 435, BOXSIZE * 2, BOXSIZE , 8, WHITE);
  tft.fillRoundRect(BOXSIZE * 6, 435, BOXSIZE * 2 , BOXSIZE , 8, BLACK);
  tft.setCursor (260, 450);
  tft.setTextSize (1);
  tft.setTextColor(WHITE);
  tft.println("PEN ");

}
void save()  {
  tft.drawRoundRect(BOXSIZE * 6 , 0 , BOXSIZE * 2 , BOXSIZE , 8 , WHITE);
  tft.fillRoundRect(BOXSIZE * 6, 0, BOXSIZE * 2 , BOXSIZE , 8, BLACK);
  tft.setCursor (260, 20);
  tft.setTextSize (1);
  tft.setTextColor(WHITE);
```

```
  tft.println("SAVE ");
}

void back() {
  tft.drawRoundRect(0, 435, BOXSIZE * 2, BOXSIZE , 8, WHITE);
  tft.fillRoundRect(0, 435, BOXSIZE * 2 , BOXSIZE , 8, BLACK);
  tft.setCursor (20, 450);
  tft.setTextSize (1);
  tft.setTextColor(WHITE);
  tft.println("BACK ");
}



//********************//
//**SD CARD INTIALISATION**//
void sdcard()
{
  Serial.begin(9600);

  tft.reset();
  tft.begin(0x9325);

  Serial.print(F("Initializing SD card..."));
  if (!SD.begin(SD_CS))
  {
    Serial.println(F("failed"));
    return;
  }
}
```

```
//**SAVING IN SD CARD**//

File bmpFile;
void writeTwo (uint16_t word) {
  bmpFile.write(word & 0xFF); bmpFile.write((word >> 8) & 0xFF);
}

// Write four bytes, least significant byte first
void writeFour (uint32_t word) {
  bmpFile.write(word & 0xFF); bmpFile.write((word >> 8) & 0xFF);
  bmpFile.write((word >> 16) & 0xFF); bmpFile.write((word >> 24) & 0xFF);
}


void bmpSave() {
  // Write two bytes, least significant byte first
  uint32_t filesize, offset;
  uint16_t width = tft.width(), height = tft.height();
  char filename[11] = "image1.bmp";
  SD.begin();
  while (SD.exists(filename)) {
    filename[5]++;
  }
  bmpFile = SD.open(filename, FILE_WRITE);
  // On error hang up
  if (!bmpFile) for (;;);
  digitalWrite(LED_BUILTIN, HIGH);
```

```
if (bmpFile) {
 // File header: 14 bytes
 bmpFile.write('B'); bmpFile.write('M');
 writeFour(14 + 40 + 12 + width * height * 2); // File size in bytes
 writeFour(0);
 writeFour(14 + 40 + 12);          // Offset to image data from start
 Serial.print("\n Writing");
 // Image header: 40 bytes
 writeFour(40);                  // Header size
 writeFour(width);                // Image width
 writeFour(height);               // Image height
 writeTwo(1);                    // Planes
 writeTwo(16);                   // Bits per pixel
 writeFour(0);                   // Compression (none)
 writeFour(0);                   // Image size (0 for uncompressed)
 writeFour(0);                   // Preferred X resolution (ignore)
 writeFour(0);                   // Preferred Y resolution (ignore)
 writeFour(0);                   // Colour map entries (ignore)
 writeFour(0);                   // Important colours (ignore)
 //
 // Colour masks: 12 bytes
 writeFour(0b0000011111100000);     // Green
 writeFour(0b1111100000000000);     // Red
 writeFour(0b0000000000011111);     // Blue
 //
 // Image data: width * height * 2 bytes
 for (int y = height - 1; y >= 0; y--) {
  for (int x = 0; x < width; x++) {
    writeTwo(tft.readPixel(x, y));
```

```
      // Each row must be a multiple of four bytes
     }
    }
    Serial.print("\n reading pixel done ");
    // Close the file
    bmpFile.close();
    digitalWrite(LED_BUILTIN, LOW);

    Serial.print(" \n done saving ");
  }
  else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
}



//*****************//
//**VOID SETUP**//

void setup(void) {

  Serial.begin(9600);
  Serial.print("Initialzing SD card...");
  pinMode(10, OUTPUT);

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
  }
```

```
Serial.println("initialization done.");
SD.mkdir("Folders");
SD.mkdir("Folders/ENGLISH");
SD.mkdir("Folders/HINDI");
SD.mkdir("Folders/MATHS");

Serial.begin(9600);
Serial.println(F("Paint!"));
tft.reset();
tft.begin(0x9341);
tft.setRotation(0);

//initial screen
tft.fillScreen(WHITE);
// Initial screen
tft.setCursor (75, 90);
tft.setTextSize (4);
tft.setTextColor(MAGENTA);
tft.println("WRITING");
tft.setCursor (130, 150);
tft.setTextSize (4);
tft.setTextColor(MAGENTA);
tft.println("PAD");
tft.setCursor (110, 250);
tft.setTextSize (1);
tft.setTextColor(BLACK);
tft.println("Touch to proceed");
currentpage = 0;
waitOneTouch();
```

```
  tft.fillScreen(BLACK);
  screen();
  pinMode(13, OUTPUT);

}



//***************//
//**VOID LOOP **//

void loop()
{
  digitalWrite(13, HIGH);
  TSPoint p = ts.getPoint();
  digitalWrite(13, LOW);
  pinMode(XM, OUTPUT);
  pinMode(YP, OUTPUT);

  if (currentpage == 0)
  {
    if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
      p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());

      // *** SPFD5408 change -- End
      p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
      if (p.y < BOXSIZE) {
        Serial.println("ENGLISH");

        tft.fillRoundRect(60, 180, 200, 40, 8, WHITE);
```

```
  delay(70);
  tft.fillRoundRect(60, 180, 200, 40, 8, RED);
  tft.drawRoundRect(60, 180, 200, 40, 8, WHITE);
  tft.setCursor(80, 250);
  tft.println("loading....");
  delay(70);
  currentpage = 1;
  english();
}


else if (p.y < BOXSIZE * 2)
{
  Serial.println("HINDI");
  currentpage = 2;
  tft.fillRoundRect(60, 180, 200, 40, 8, WHITE);
  delay(70);

  tft.fillRoundRect(60, 180, 200, 40, 8, RED);
  tft.drawRoundRect(60, 180, 200, 40, 8, WHITE);

  tft.setCursor(80, 250);
  tft.print("loading....");
  delay(70);

  hindi();
}

else  if (p.y < BOXSIZE * 3)
```

```
    {
     Serial.println("MATHS");

     currentpage = 3;

     tft.fillRoundRect(60, 180, 200, 40, 8, WHITE);
     delay(70);

     tft.fillRoundRect(60, 180, 200, 40, 8, RED);
     tft.drawRoundRect(60, 180, 200, 40, 8, WHITE);

     tft.setCursor(80, 250);
     tft.print("loading....");
     delay(70);

     maths();
    }
    else if (p.y > BOXSIZE * 3)
    { screen();
    }
  }
}

if (currentpage == 1)
{
  if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
    p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
    p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
```

```
 if ((p.y > 435) && (p.x < BOXSIZE * 2 ) && (p.x > 0))
 { screen();
  currentpage = 0; // BACK
 }
 else if ((p.y > 435) && (p.x < BOXSIZE * 8 ) && (p.x > BOXSIZE * 6))
 {
  currentcolor = BLUE;
  //PEN
 }
 else if ((p.x < BOXSIZE * 5) && (p.x > BOXSIZE * 3) && (p.y > 435))
 { currentcolor = WHITE;
  //ERASER
 }
 else if ((p.y > 0) && (p.y < 40) && ( p.x < BOXSIZE * 8) && ( p.x > BOXSIZE * 6))
 {
  bmpSave(); //save
  tft.setCursor (120, 420);
  tft.setTextSize (1);
  tft.setTextColor(RED);
  tft.println("DONE SAVING");
  delay(1000);
  currentpage = 1;
  english();

 }

}

if (((p.y - PENRADIUS) > 0) && ((p.y + PENRADIUS) < tft.height() - 45)) {
```

```
    tft.fillCircle(p.x, p.y, PENRADIUS, currentcolor);   //(height -45 since we are using y = 435 so
480 - 435 = 45 , pen will not draw in that area)
   }
  }



 if (currentpage == 2)
 {
  if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
    p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
    p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());

    if ((p.y > 435) && (p.x < BOXSIZE * 2 ) && (p.x > 0))
    {
     screen();
     currentpage = 0; // BACK
    }
    else if ((p.y > 435) && (p.x < BOXSIZE * 8 ) && (p.x > BOXSIZE * 6))
    {
     currentcolor = BLUE;
     //PEN
    }
    else if ((p.x < BOXSIZE * 5) && (p.x > BOXSIZE * 3) && (p.y > 435))
    {
     currentcolor = WHITE;
     //ERASER
    }
    else if ((p.y > 0) && (p.y < 40) && ( p.x < BOXSIZE * 8) && ( p.x > BOXSIZE * 6))
    {
```

```
    bmpSave();//save
    tft.setCursor (120, 420);
    tft.setTextSize (1);
    tft.setTextColor(RED);
    tft.println("DONE SAVING");
    delay(1000);
    currentpage = 2;
    hindi();
   }

  }

  if (((p.y - PENRADIUS) > 0) && ((p.y + PENRADIUS) < tft.height() - 45)) {
   tft.fillCircle(p.x, p.y, PENRADIUS, currentcolor);
  }



}

if (currentpage == 3)
{
  if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
   p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
   p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());

   if ((p.y > 435) && (p.x < BOXSIZE * 2 ) && (p.x > 0))
   {
    screen();
    currentpage = 0; // BACK
```

```
    }
    else if ((p.y > 435) && (p.x < BOXSIZE * 8 ) && (p.x > BOXSIZE * 6))
    {
     currentcolor = BLUE;
     //PEN
    }
    else if ((p.x < BOXSIZE * 5) && (p.x > BOXSIZE * 3) && (p.y > 435))
    {
     currentcolor = WHITE;
     //ERASER
    }
    else if ((p.y > 0) && (p.y < 40) && ( p.x < BOXSIZE * 8) && ( p.x > BOXSIZE * 6))
    {
     bmpSave();//save
     tft.setCursor (120, 420);
     tft.setTextSize (1);
     tft.setTextColor(RED);
     tft.println("DONE SAVING");
     delay(1000);
     currentpage = 3;
     maths();
    }
   }
   if (((p.y - PENRADIUS) > 0) && ((p.y + PENRADIUS) < tft.height() - 45)) {
    tft.fillCircle(p.x, p.y, PENRADIUS, currentcolor);
   }
  }
}
```

APPENDIX 2

References

[1] https://www.mydimmerswitch.com/how-tos/what-color-light-is-best-for-your-eyes/

[2] https://www.rs-online.com/designspark/what-is-arduino-uno-a-getting-started-guide#:~:text=Arduino%20UNO%20is%20a%20low,and%20motors%20as%20an%20output.

[3] https://www.engineersgarage.com/arduino-ili9486-driver-3-5-inch-tft-lcd-touch-screen/

[4] https://docs.arduino.cc/tutorials/uno-rev3/intro-to-board

[5] (https://www.arduino.cc/en/Tutorial/Foundations/Memory)

[6] (https://www.orientdisplay.com/knowledge-base/tft-basics/lcd-history/#:~:text=The%20TFT%20LCD%20was%20then,the%20pixels%20could%20be%20displayed)

[7]https://www.jdisplay.com/english/technology/lcdbasic.html#:~:text=A%20change%20in%20voltage%20applied,to%20produce%20full%2Dcolor%20images.

[8] (http://www.lcdwiki.com/3.5inch_Arduino_Display-UNO)

[9] https://thesolaruniverse.wordpress.com/2017/02/10/arduino-test-bench-with-a-3-5-inch-tft-displaying-in-analog-fashion/

[10] https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html

[11] https://learn.adafruit.com/adafruit-2-8-tft-touch-shield-v2/adafruit-gfx-library

[12] https://forum.arduino.cc/t/arduino-jpeg-and-bmp-library-with-sram-shield/380727

[13]http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/2003_w/misc/bmp_file_format/bmp_file_format.htm

[14] Dr.P.Arun**.** “Serial Communication”, Serial Communication Lecture, BSc(H) Electronics, Shree Guru Tegh Bahadur Khalsa College, February, 2022.