Final Project Write up

CS 207

Roman Yushchyk

200368308

Stargate Alarm Clock

Introduction:

My motivation is to wake up to school on time. My current alarm clock has served me a whole eternity by the technological standards. Almost six years of continues work without a need to even switch batteries. Perhaps because of the solar penal, perhaps because it come from a dollar store it was my best friend for a longer period of time. On the negative side it has no touchscreen or an option to change the alarm sound, and to top it all up, a few weeks ago my background light ceased to work. What deems it almost non-useable at night. This was the moment I decided to build my own super alarm clock instead of getting something widely available on the market.

On the end of the day, when this project is accomplished, I will be able to add or modify any of the functions or buttons in a menu and that's cost more than any alarm clock ever made before me.

Background

This project was inspired by one similar project on internet. You can find it by following this link.

I also borrow a few pictures to make an example from the website above.

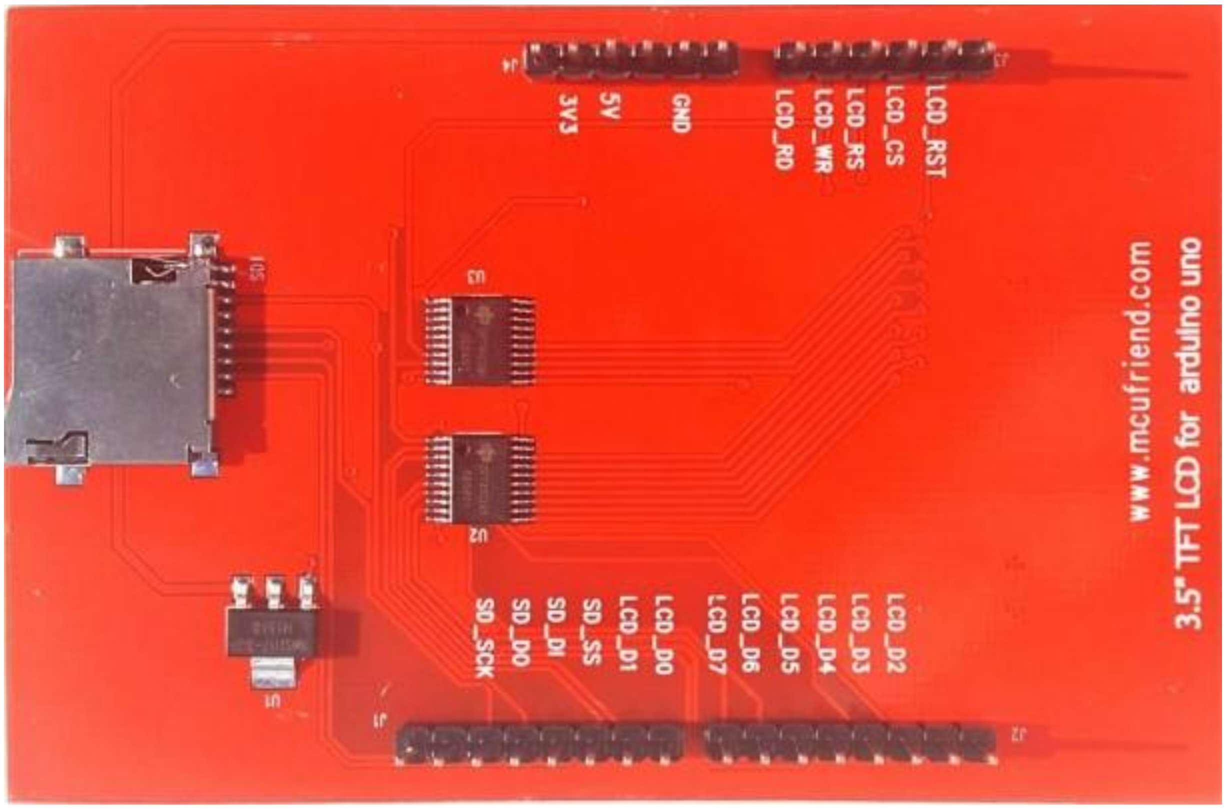
Rather than going with a poorly designed display I choose to replace it with a more compact mcufriend touchscreen. This leads to a change of library and I have to code the whole alarm clock and a user interface all by myself. I started with a similar layout for the code and modified it as I go.

Motivation: I wanted to create a perfect alarm clock with all of the bells and lights.

Description of your design process

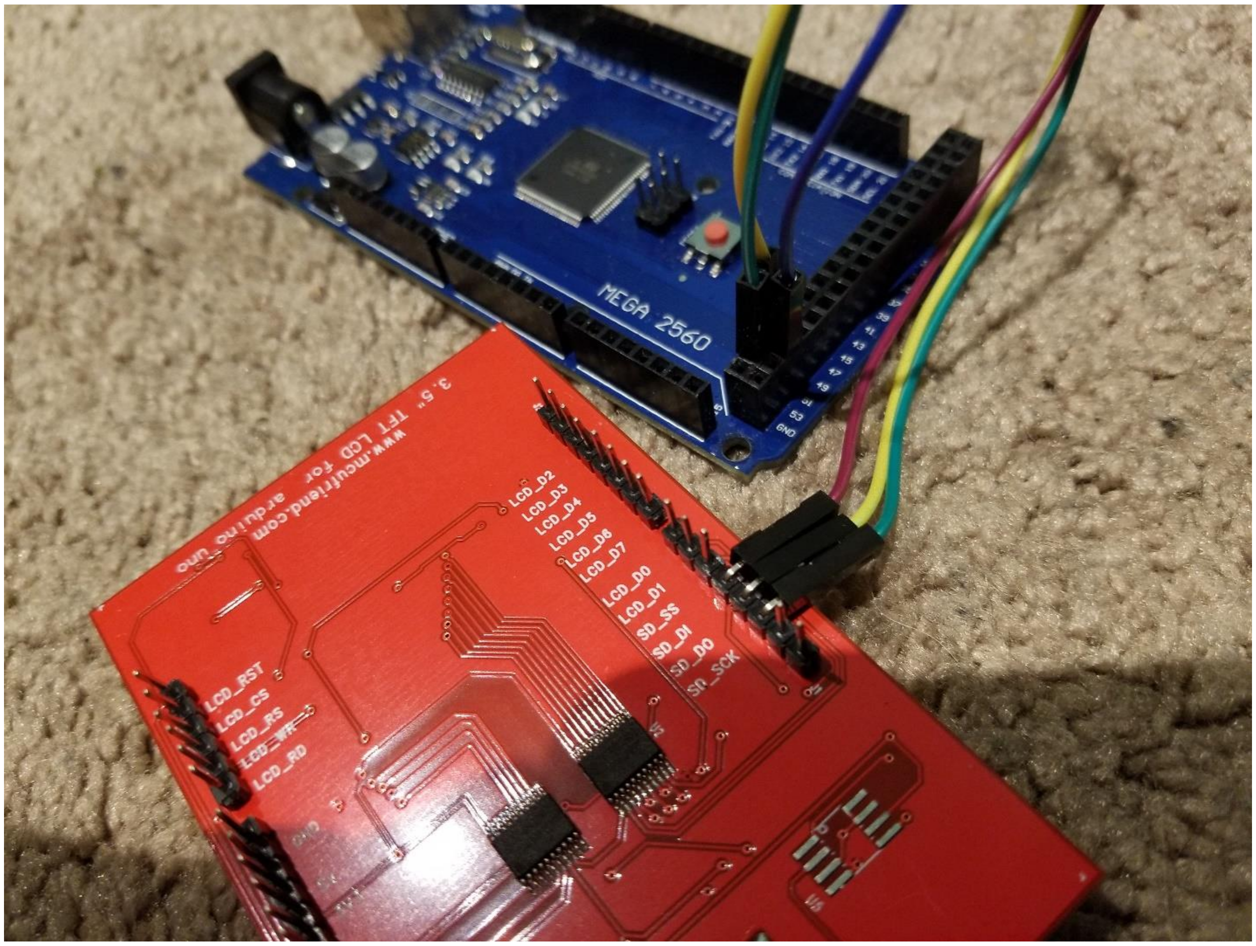
1) First of one we need to copy libraries from this repository to your computer.
2) Second step is to band pins 11, 12, 13 to 90 degrees to allow faster download speed trough native pins 50, 51, 52 on Arduino atMega 2560.

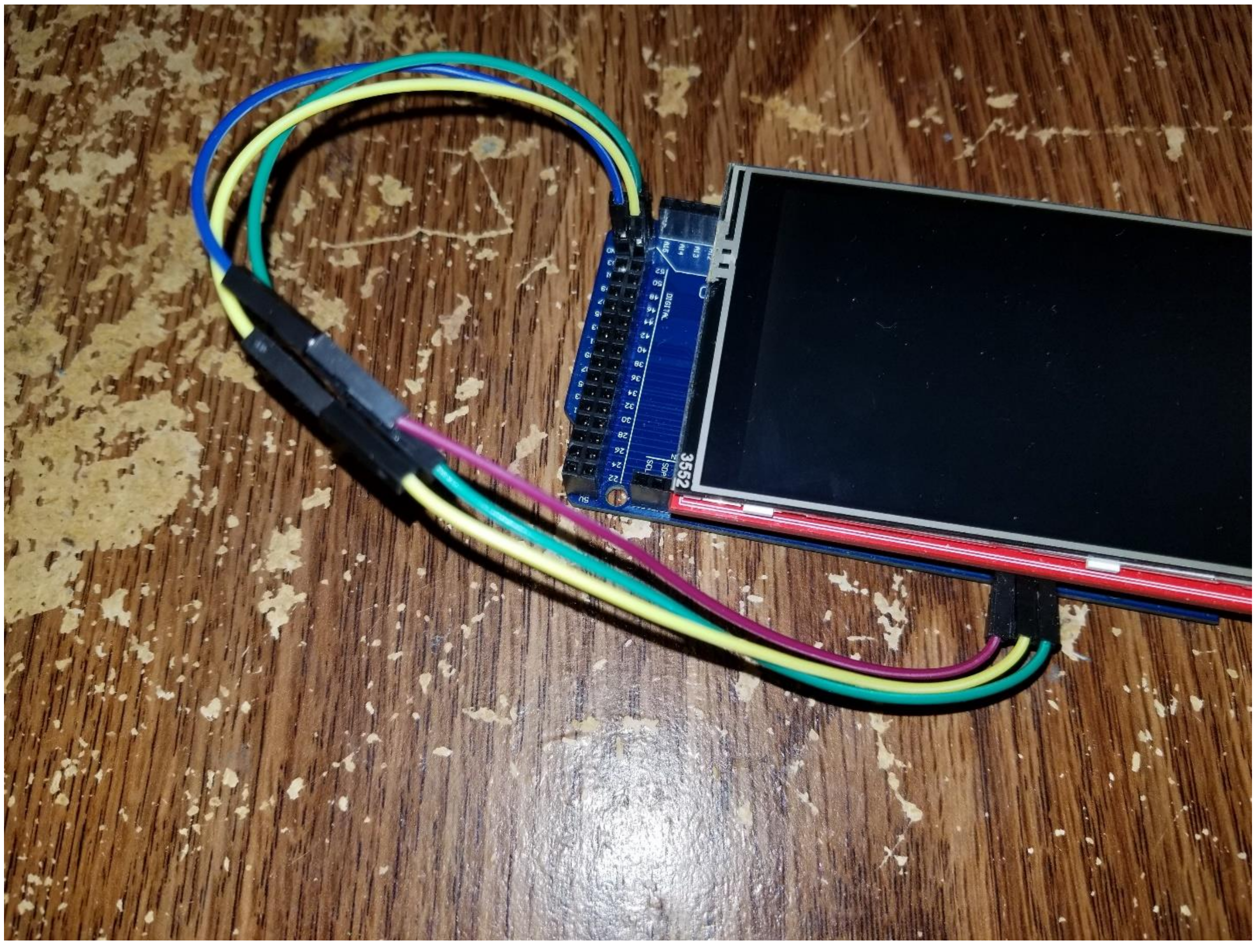
Here how it looks from below before bending pins on this TFT.



Pins marked **SD_SCK (Clock)** – pin 50(mega2560), **SD_D0**(Data out) – pin 52 (Mega2560) and **SD_D1**(Data in) – pin 51(Mega 2560) pins need to banded at 90 degrees to enable native SPI on Arduino Mega2560. Simply run jumpers to each pin accordingly.

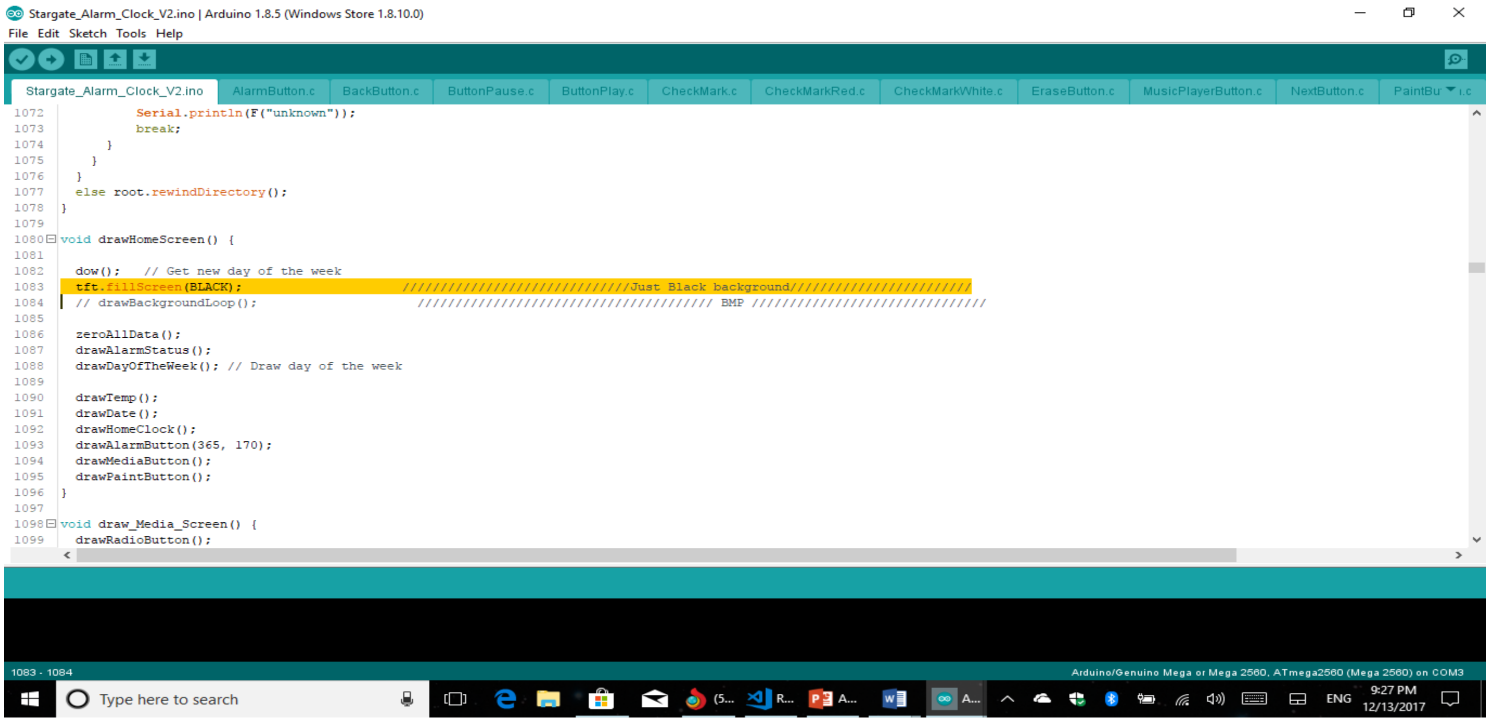
Here is how new TFT SD card wiring looks like after doing steps above.





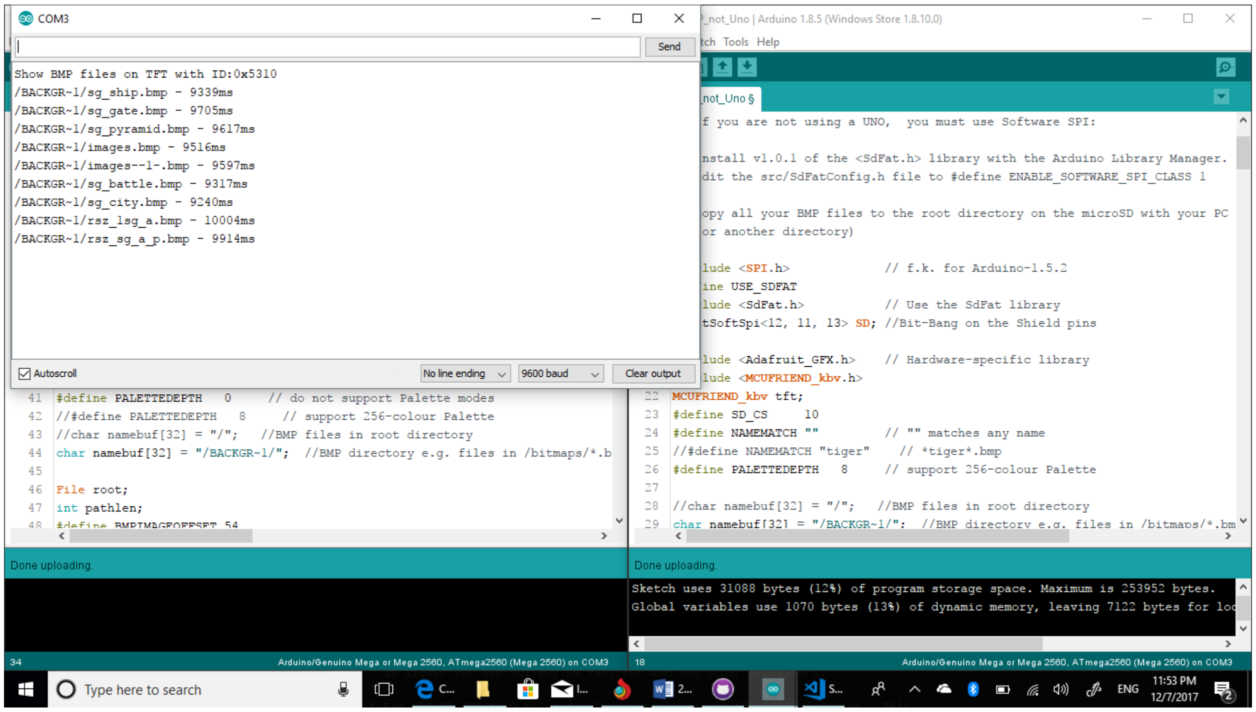
By using SDFat library rather than SD and Software Serial instead of SPI we increased loading speed by a factor of two. It dropped from an average of 10 seconds for 480 X 320 bmp to half of it, just 5 seconds.

SDFat load time is only 5 seconds for an average background. You can change drawHomeScreen() function. Just comment it out for or debagging purpose. Simply uncomment one and comment other one with inn.



```
          Serial.println(F("unknown"));
          break;
      }
    }
  }
  else root.rewindDirectory();
}

void drawHomeScreen() {

  dow();   // Get new day of the week
  tft.fillScreen(BLACK);          ///////////////////////////////Just Black background///////////////////////
  // drawBackgroundLoop();          ////////////////////////////////////// BMP ///////////////////////////////

  zeroAllData();
  drawAlarmStatus();
  drawDayOfTheWeek(); // Draw day of the week

  drawTemp();
  drawDate();
  drawHomeClock();
  drawAlarmButton(365, 170);
  drawMediaButton();
  drawPaintButton();
}

void draw_Media_Screen() {
  drawRadioButton();
```

```
1083 - 1084                                   Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3
```

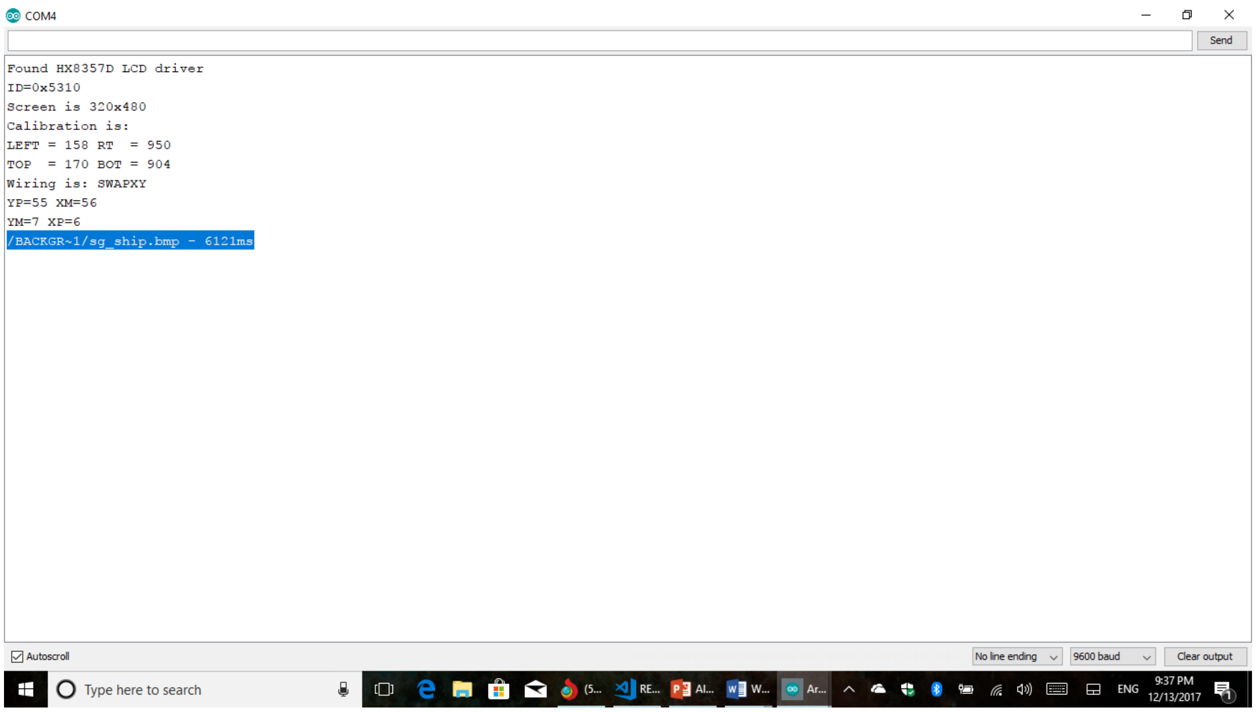

```
Show BMP files on TFT with ID:0x5310
/BACKGR~1/sg_ship.bmp - 9339ms
/BACKGR~1/sg_gate.bmp - 9705ms
/BACKGR~1/sg_pyramid.bmp - 9617ms
/BACKGR~1/images.bmp - 9516ms
/BACKGR~1/images--1-.bmp - 9597ms
/BACKGR~1/sg_battle.bmp - 9317ms
/BACKGR~1/sg_city.bmp - 9240ms
/BACKGR~1/rsz_1sg_a.bmp - 10004ms
/BACKGR~1/rsz_sg_a_p.bmp - 9914ms
```

```
#define PALETTEDEPTH   0     // do not support Palette modes
//#define PALETTEDEPTH   8     // support 256-colour Palette
//char namebuf[32] = "/";   //BMP files in root directory
char namebuf[32] = "/BACKGR~1/";  //BMP directory e.g. files in /bitmaps/*.b

File root;
int pathlen;
#define BMPIMAGEOFFSET 54
```

```
f you are not using a UNO,  you must use Software SPI:

nstall v1.0.1 of the <SdFat.h> library with the Arduino Library Manager.
dit the src/SdFatConfig.h file to #define ENABLE_SOFTWARE_SPI_CLASS 1

opy all your BMP files to the root directory on the microSD with your PC
or another directory)

lude <SPI.h>             // f.k. for Arduino-1.5.2
ine USE_SDFAT
lude <SdFat.h>           // Use the SdFat library
tSoftSpi<12, 11, 13> SD; //Bit-Bang on the Shield pins

lude <Adafruit_GFX.h>    // Hardware-specific library
lude <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;
#define SD_CS     10
#define NAMEMATCH ""         // "" matches any name
//#define NAMEMATCH "tiger"    // *tiger*.bmp
#define PALETTEDEPTH   8     // support 256-colour Palette

//char namebuf[32] = "/";   //BMP files in root directory
char namebuf[32] = "/BACKGR~1/";  //BMP directory e.g. files in /bitmaps/*.bm
```

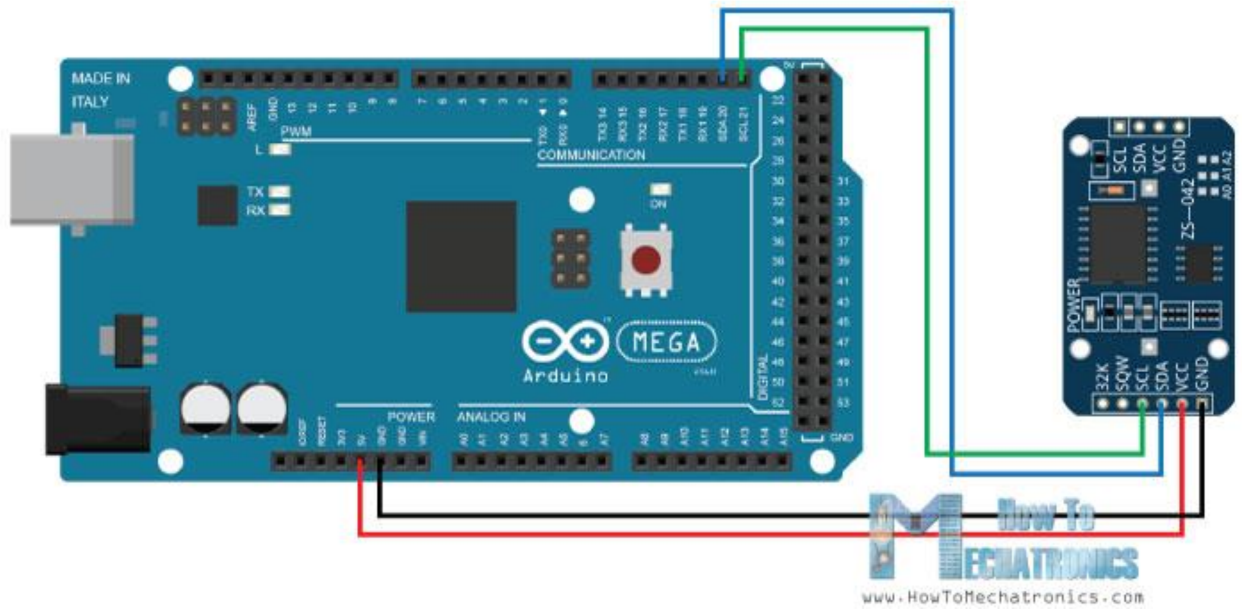
```
Done uploading.
Sketch uses 31088 bytes (12%) of program storage space. Maximum is 253952 bytes.
Global variables use 1070 bytes (13%) of dynamic memory, leaving 7122 bytes for loc
```

Upload time for a picture without a hardware modification.

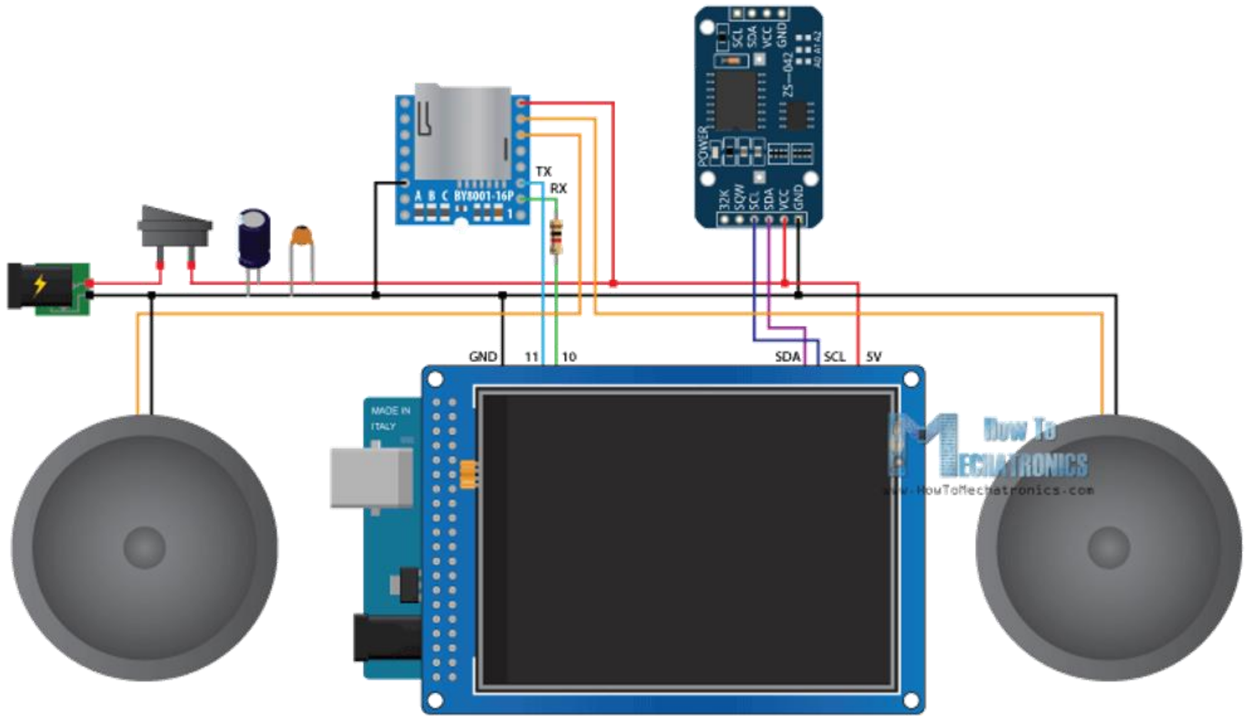

```
Found HX8357D LCD driver
ID=0x5310
Screen is 320x480
Calibration is:
LEFT = 158 RT  = 950
TOP  = 170 BOT = 904
Wiring is: SWAPXY
YP=55 XM=56
YM=7 XP=6
/BACKGR~1/sg_ship.bmp - 6121ms
```

Upload time for an integrated code with a background picture.

Let’s take a look at the RTC clock. RTC connection did not changed and you can set a clock and date from the setup loop.



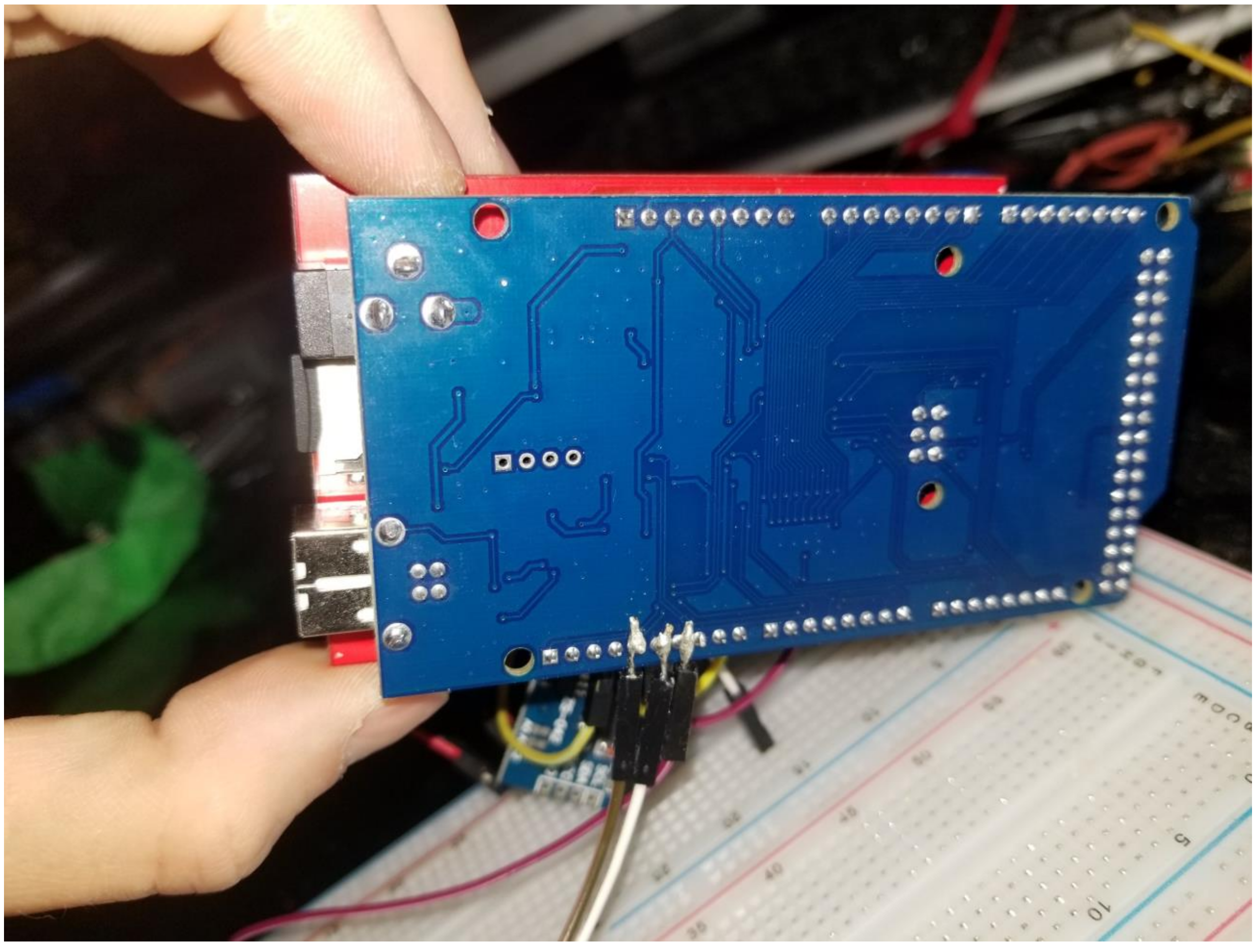
Wiring for a MP3 player has changed a little. I'm using pins 13, 12, which was made free by the SD card bypass.

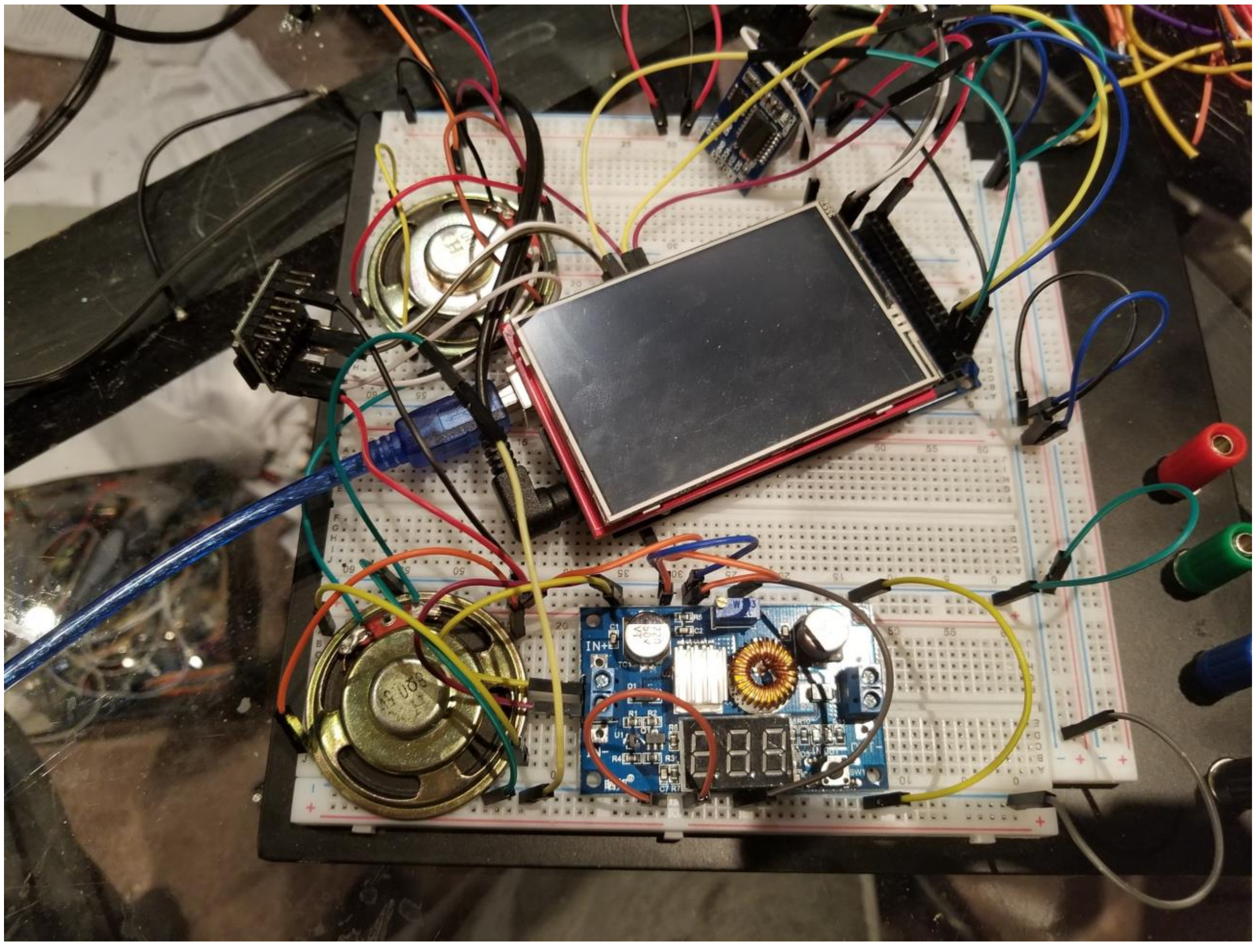


To make it compact I decided to solder my wires directly onto my mega. To prevent lose ends while I use my alarm.

Mercury tilt switch is connected to the pin number 11 in my case.

Here is a better view of the Mp3 13, 12 connection and A tilt switch at the pin 11.





General overlay pictures

Description of your design process

I started with a basic layout from the mechatronics website and begin to change the code to feet a new libraries and hardware. Slowly by slowly I got to a completely custom-building code. I even start modifying LIBRIS when I could not make it to work. Nut one problem at a time. One dislocated pixel at a time I was able to finish main requirements. Main menu, Alarm and a media player screen. I added a few colors to the paint program and also add days of the week selection for the Alarm. What was a strong feed on its own. Things break and come apart. I spend more than a week worth of working at night until 5 AM to make it work perfectly fine and the way I design it.

Code Logic.

Add library

Copy icons .C files from images folder to the sketch folder in your Arduino

Comply and upload

If you got all the basic libraries' and those that I provide you should get a good mini-OS.

Description of how the project works (i.e. a user manual of sorts)

First of one you want to set your RTC to a time and Date Trough Setup Loop.

Second step is to set alarms to the same mode 12 or 24 hrs. trough true/false statement in a Setup.

Both of them please if you choose another setup than 12-hour day. AM PM is a default.

Further you shall use only interface of an alarm clock to set new alarms.

Most of the coding delay was ceased solely by a super complex screen control with many multiple buttons that do a double function accordingly to a state of other selectors for each button.

Code get complicated and jumps trough created void statements when it needs it. From an example to update seconds every time. You need to call this function call from a loop. When you want to do it once or to open another screen only if other conditions is meat. It is all adds up to a complexity.

Memory

Static Random-Access Memory SRAM is at a max 4kb because of PROGMEEM use for .C files.

Which holds all the Icons for a maximum speed.

If you want to add more graphics than use BMP and SDfat to draw a bigger graphical object like a background.

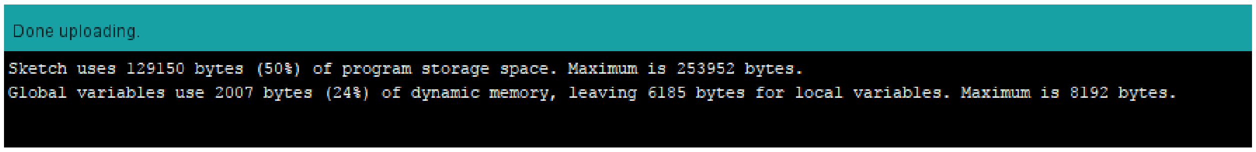
SD Card in mp3 player has to have at least one mp3 or wav file.

SD Card inside the touchscreen Background folder and 480X320 .bmp files.

Now you are ready to run this space ship.

Sketch memory is at 50%

Global variables is at 24%

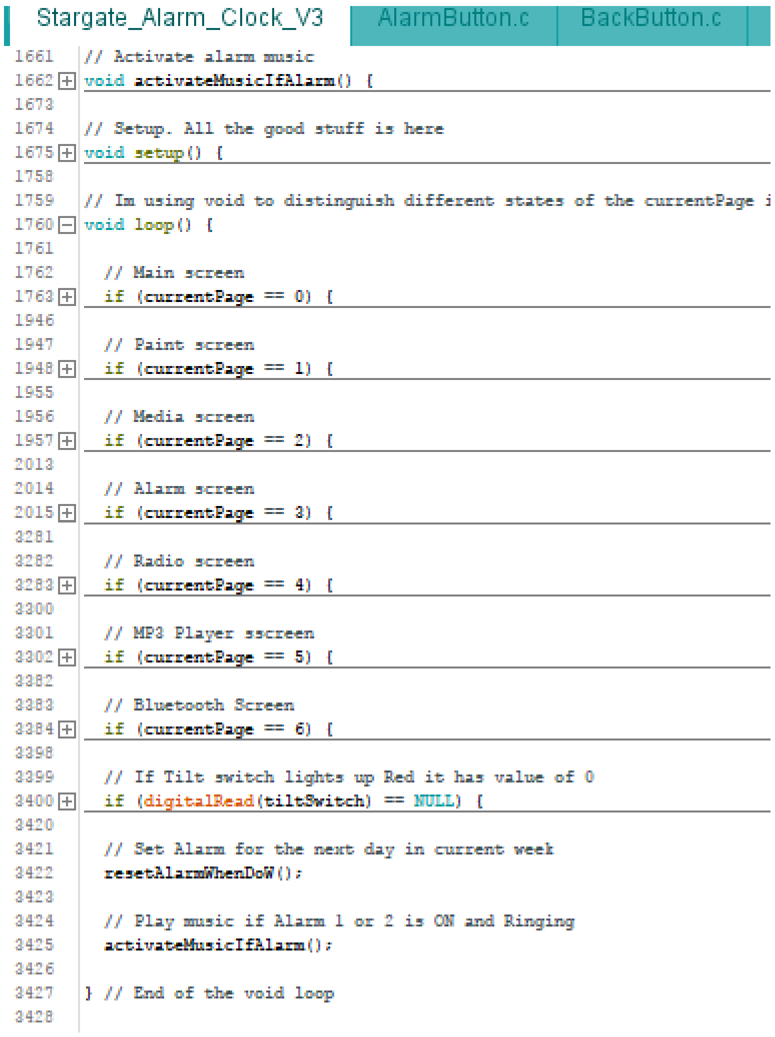


```
Done uploading.
Sketch uses 129150 bytes (50%) of program storage space. Maximum is 253952 bytes.
Global variables use 2007 bytes (24%) of dynamic memory, leaving 6185 bytes for local variables. Maximum is 8192 bytes.
```

Let’s move on to the code logic.

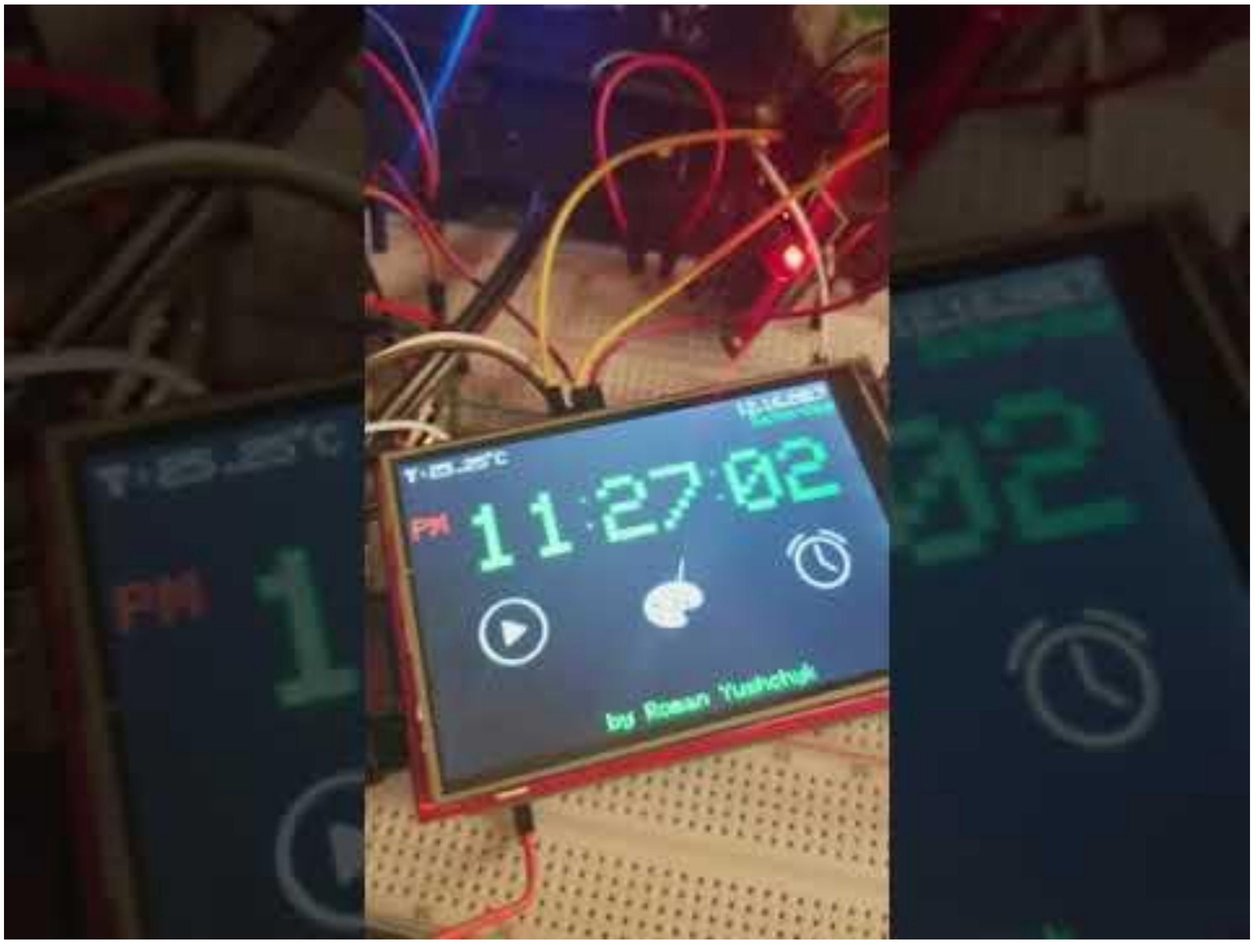
This is the loop with 6 different screens and reset alarm for the next week and play alarm functions.

Each of those if statements contain all of the buttons and corresponding actions for each individual screen. We have to change screens manually assigning integer value to currentPage Each time we press a back button or click a radio button. Explaining each line of code would take simply too much time. So, I try to generalize to the point you can understand the whole idea behind this sketch.



```
// Activate alarm music
void activateMusicIfAlarm() {

// Setup. All the good stuff is here
void setup() {

// Im using void to distinguish different states of the currentPage i
void loop() {

  // Main screen
  if (currentPage == 0) {

  // Paint screen
  if (currentPage == 1) {

  // Media screen
  if (currentPage == 2) {

  // Alarm screen
  if (currentPage == 3) {

  // Radio screen
  if (currentPage == 4) {

  // MP3 Player sscreen
  if (currentPage == 5) {

  // Bluetooth Screen
  if (currentPage == 6) {

  // If Tilt switch lights up Red it has value of 0
  if (digitalRead(tiltSwitch) == NULL) {

  // Set Alarm for the next day in current week
  resetAlarmWhenDoW();

  // Play music if Alarm 1 or 2 is ON and Ringing
  activateMusicIfAlarm();

} // End of the void loop
```

This is how everything looks together in one video.



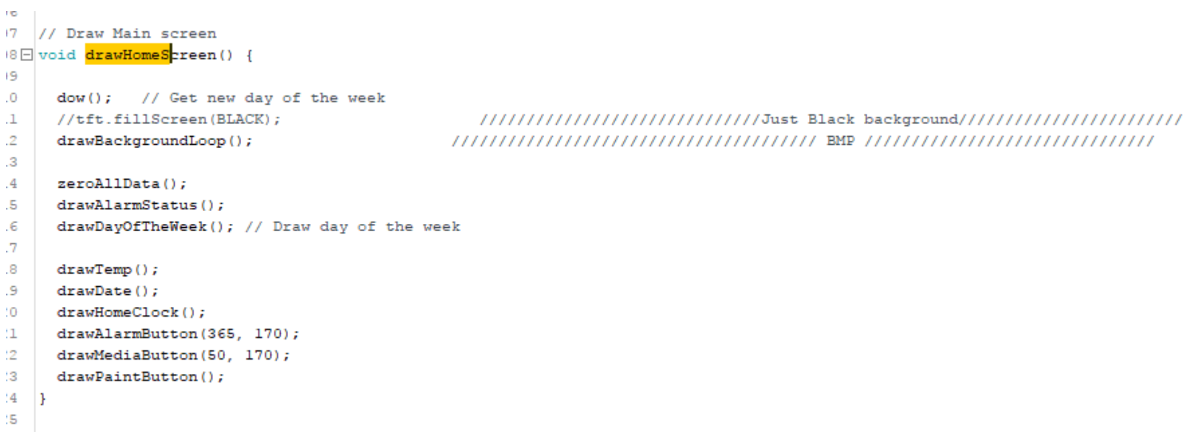
Let’s begin from a Main screen

First, we set current Page = 0 in a setup to boot in a main screen.

Than we draw void called drawHomeScreen() which will draw mian sceen buttons and layout.

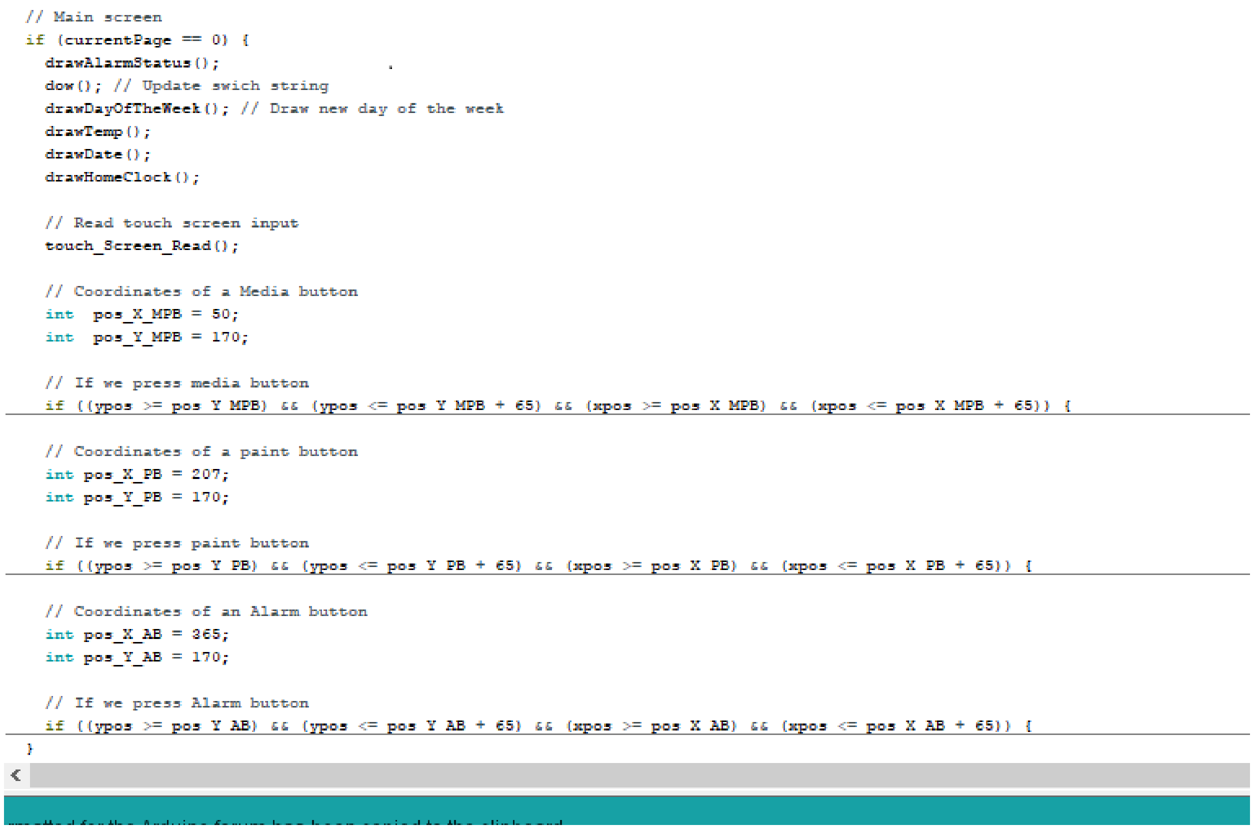
But I am checking for a change of time and temperatiure from a loop (if currentPage = 0) than I read touchscreen and wait for user input while updating time and temp on all of the screens.

Example



```
// Draw Main screen
void drawHomeScreen() {

  dow();   // Get new day of the week
  //tft.fillScreen(BLACK);                    ////////////////////////////////Just Black background////////////////////////
  drawBackgroundLoop();                    /////////////////////////////////////// BMP ////////////////////////////////

  zeroAllData();
  drawAlarmStatus();
  drawDayOfTheWeek(); // Draw day of the week

  drawTemp();
  drawDate();
  drawHomeClock();
  drawAlarmButton(365, 170);
  drawMediaButton(50, 170);
  drawPaintButton();
}
```

And a loop equivalent will be:



```
// Main screen
if (currentPage == 0) {
  drawAlarmStatus();
  dow(); // Update swich string
  drawDayOfTheWeek(); // Draw new day of the week
  drawTemp();
  drawDate();
  drawHomeClock();

  // Read touch screen input
  touch_Screen_Read();

  // Coordinates of a Media button
  int  pos_X_MPB = 50;
  int  pos_Y_MPB = 170;

  // If we press media button
  if ((ypos >= pos_Y_MPB) && (ypos <= pos_Y_MPB + 65) && (xpos >= pos_X_MPB) && (xpos <= pos_X_MPB + 65)) {

  // Coordinates of a paint button
  int pos_X_PB = 207;
  int pos_Y_PB = 170;

  // If we press paint button
  if ((ypos >= pos_Y_PB) && (ypos <= pos_Y_PB + 65) && (xpos >= pos_X_PB) && (xpos <= pos_X_PB + 65)) {

  // Coordinates of an Alarm button
  int pos_X_AB = 365;
  int pos_Y_AB = 170;

  // If we press Alarm button
  if ((ypos >= pos_Y_AB) && (ypos <= pos_Y_AB + 65) && (xpos >= pos_X_AB) && (xpos <= pos_X_AB + 65)) {
}
```

Where each if statement is a button with its own instructions.

Example of button bush in a loop page 0.

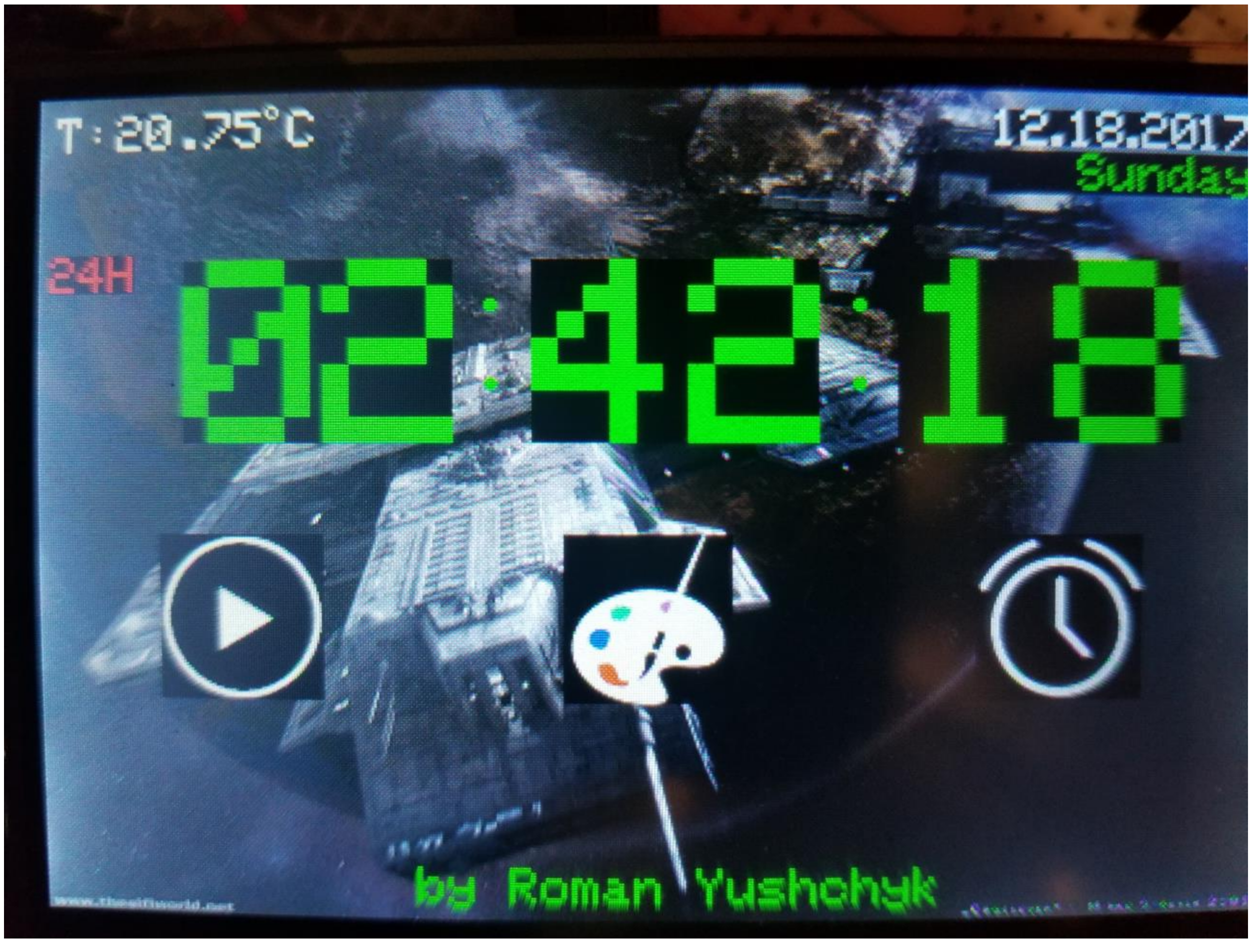
```
// Coordinates of a Media button
int  pos_X_MPB = 50;
int  pos_Y_MPB = 170;

// If we press media button
if ((ypos >= pos_Y_MPB) && (ypos <= pos_Y_MPB + 65) && (xpos >= pos_X_MPB) && (xpos <= pos_X_MPB + 65)) {
  // Zero all data is used in a next screen
  zeroAllData();
  // Set sceren black
  tft.fillScreen(BLACK); // Sets the background color of the area where the text will be printed to black
  // Change scren count
  draw_Media_Screen();
  currentPage = 2;
}
```

Same logic applies trough out the whole code.

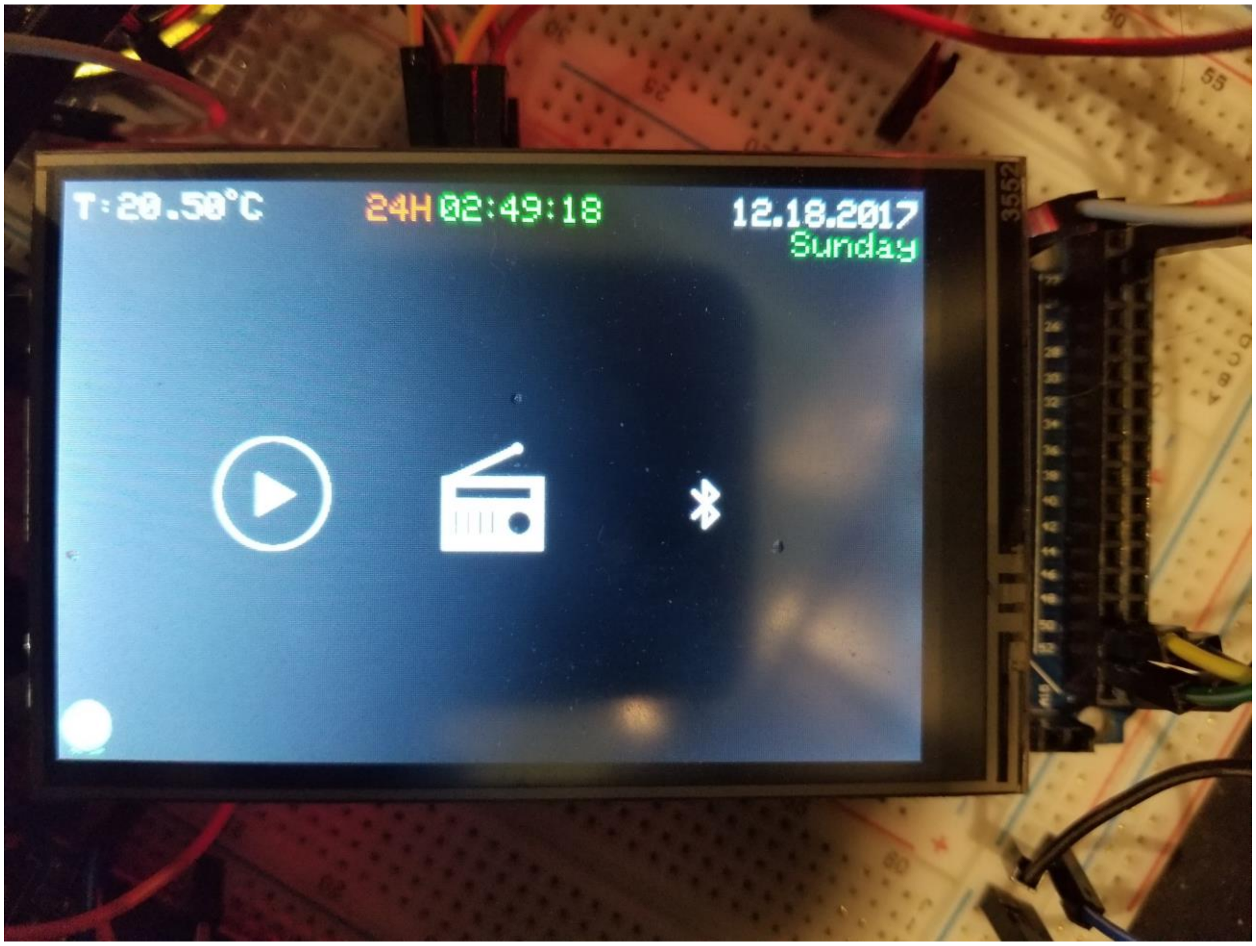
And it is easy to follow it if you know some simple secret. A1h12 true for 12 hour format and aA1PM true for PM. That's should help with alarm clock navigation.

This is the result of those two paces of code I decide to include.

Before:



After:



Same rules just repeat many time for each of the six current screens.

Original values for alarm are without a attachment new. They are retrieved from RTC.

A1 & newA1

Then question is to map numbers properly from an arrays and into a new day of the week.

I try to name all my function in such a way that they are self-explanatory and easy to read.

When we tilt our sensor I just sending a mp3.puase() and to play I use mp3.play()

drawTemp() for example is used at every screen and it is updating value of the temperature if that has any change. That's why we need to zero all data before drawing a new screen

I would strongly suggest to open each library and look over the available commands before following with the code reading.

Discussion of my milestones

I did everything from my list and even more. I add extra screens. Modify paint colors and even add background to the whole clock. I did make mercury tilt switch to work the way I want but there were a few things that I will take care after the final. This is Bluetooth, FM, radio, Wireless phone charger and a WIFI module. As well as building a 3D printed case for it all. Thanks Lord I have my own 3D printer.

Appendix A: Code

It is a rather large code.

My GitHub name: yushchyr

https://github.com/yushchyr/CS207/tree/master/CS207_Alarm_Clock_Project

My code as it is on gut Hub:

```
[code]
/*
  Roman Yushchyk
  200368308
  Alarm clock
*/



// Intialisation
#include <EEPROM.h>
#include <Wire.h>
#include <BY8001.h>
#include <DS3231.h>



// TFT Initiatioon
#include <SoftwareSerial.h>
#include <Adafruit_GFX.h>    // Core graphics library
#include <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;       // hard-wired for UNO shields anyway.
#include <TouchScreen.h>
#if defined(__SAM3X8E__)
#undef __FlashStringHelper::F(string_literal)
```

```
#define F(string_literal) string_literal

#endif


// TiltSwitch pin

int tiltSwitch = 11;


// SD Initiation

#include <BlockDriver.h>

#include <FreeStack.h>

#include <MinimumSerial.h>

#include <SdFat.h>

#include <SdFatConfig.h>

#include <SysCall.h>

#define SD_CS     10

#include <SPI.h>             // f.k. for Arduino-1.5.2

#define USE_SDFAT

SdFat SD; // Bit-Bang on the Shield pins



// BMP Background

#define NAMEMATCH "" // "" matches any name

//#define NAMEMATCH "tiger"   // *tiger*.bmp

#define PALETTEDEPTH   0     // do not support Palette modes

//#define PALETTEDEPTH   8     // support 256-colour Palette

//char namebuf[32] = "/";   //BMP files in root directory

char namebuf[32] = "/BACKGR~1/";  //BMP directory e.g. files in /bitmaps/*.bmp


// Part of the BMP drawing process

File root;

int pathlen;

#define BMPIMAGEOFFSET 54

#define BUFFPIXEL      20
```

```
// TFT Shield pinout

uint8_t YP = A1;  // must be an analog pin, use "An" notation!

uint8_t XM = A2;  // must be an analog pin, use "An" notation!

uint8_t YM = 7;   // can be a digital pin

uint8_t XP = 6;   // can be a digital pin

uint8_t SwapXY = 0;


// TFT Shield corner touch points values

uint16_t TS_LEFT = 961;

uint16_t TS_RT  = 134;

uint16_t TS_TOP = 917;

uint16_t TS_BOT = 120;

volatile uint16_t xpos = 0, ypos = 0;  //screen coordinates

char *name = "Unknown controller";

// For better pressure precision, we need to know the resistance

// between X+ and X- Use any multimeter to read it

// For the one we're using, its 300 ohms across the X plate


// Touch screen initialization

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);


// Touch point initialization

TSPoint tp;


// Minimum and maximum pressure input range

#define MINPRESSURE 20

#define MAXPRESSURE 1000


// Swap function prototype

#define SWAP(a, b) {uint16_t tmp = a; a = b; b = tmp;}


// Screen  variables

int16_t BOXSIZE;

int16_t PENRADIUS = 3;
```

```
uint16_t identifier, oldcolor, currentcolor;

uint8_t Orientation = 1;    // Landscape orientaion is default

char currentPage, playBackStatus;

String alarmString = "";




// Assign human-readable names to some common 16-bit color values: I add more colors to the paint
setup. You can choose any of this.

#define BLACK          0x0000      /*   0,   0,   0 */

#define Navy           0x000F      /*   0,   0, 128 */

#define DarkGreen      0x03E0      /*   0, 128,   0 */

#define DarkCyan       0x03EF      /*   0, 128, 128 */

#define Maroon         0x7800      /* 128,   0,   0 */

#define PURPULE        0x780F      /* 128,   0, 128 */

#define Olive          0x7BE0      /* 128, 128,   0 */

#define LightGrey      0xC618      /* 192, 192, 192 */

#define DarkGrey       0x7BEF      /* 128, 128, 128 */

#define BLUE           0x001F      /*   0,   0, 255 */

#define GREEN          0x07E0      /*   0, 255,   0 */

#define CYAN           0x07FF      /*   0, 255, 255 */

#define RED            0xF800      /* 255,   0,   0 */

#define MAGENTA        0xF81F      /* 255,   0, 255 */

#define YELLOW         0xFFE0      /* 255, 255,   0 */

#define WHITE          0xFFFF      /* 255, 255, 255 */

#define ORANGE         0xFD20      /* 255, 165,   0 */

#define GreenYellow    0xAFE5      /* 173, 255,  47 */

#define PINK           0xF81F      /**/


// Real Time Clock instance

DS3231 rtc;

bool Century = false;

byte DoW = -1;

byte oldDoW = -1;

String day_Of_The_Week = "";
```

```
bool h12;
bool PM;
byte currentDate = -1;
byte currentHours = -1;
byte currentMinutes = -1;
byte currentSeconds = -1;
float temperature = -1;

// Alarm One
byte A1Day, A1Hour, A1Minute, A1Second, A1Bits;
bool A1Dy, A1h12, A1PM;
bool alarmOneWeek[7] = {false, false, false, false, false, false, false};
bool newAlarmOne = false;
int newA1Day = -1;
int newA1Hour = -1;
int newA1Minute = -1;
bool newA1Dy = -1;
int newA1Date = -1;
bool newA1h12, newA1PM;
bool newHourSelector = false; // Selector switches to control alarm screen
bool newMinuteSelector = false;
bool newDoWSelector = false;

// Alarm Two
byte A2Day, A2Hour, A2Minute, A2Bits;
bool A2Dy, A2h12, A2PM;
bool alarmTwoWeek[7] = {false, false, false, false, false, false, false};
bool newA2h12, newA2PM;
bool newAlarmTwo = false;
int newA2Hour = -1;
int newA2Minute = -1;
bool newA2Dy = -1;
int newA2Date = -1;
int newA2Day = -1;
```

```
bool newHour2Selector = false; // Selector switches to control alarm screen

bool newMinute2Selector = false;

bool newDoW2Selector = false;




// eeprom

int eeAddressAlarmOne = 0; // random(0, 2047); // Half

int eeAddressAlarmTwo = 10; // random(2048, 4096); //  Other half


// Common start points for a Graphic block of elements

int pos_X; // Home clock

int pos_Y;

int X_A1 = 60; // Alarm 1

int Y_A1 = 80;

int X_A2 = 316; // Alarm 2

int Y_A2 = 80;


// Delay time for checkmarks

int t = 100;


// MP3 Declaration

SoftwareSerial mp3Serial(13, 12);  // RX, TX

BY8001 mp3;  // creating an instance of class BY8001 and call it 'mp3'

int mp3Folder = 0; // Folder number

int mp3Song = 1; // Song Number

int vol = 15; // Volume integer


// Show Serial info Screen

void show_Serial(void) {

  Serial.print(F("Found "));

  Serial.print(name);

  Serial.println(F(" LCD driver"));

  Serial.print(F("ID=0x"));

  Serial.println(identifier, HEX);
  Serial.println("Screen is " + String(tft.width()) + "x" + String(tft.height()));
  Serial.println("Calibration is: ");
  Serial.println("LEFT = " + String(TS_LEFT) + " RT  = " + String(TS_RT));
  Serial.println("TOP  = " + String(TS_TOP)  + " BOT = " + String(TS_BOT));
  Serial.print("Wiring is: ");
  Serial.println(SwapXY ? "SWAPXY" : "PORTRAIT");
  Serial.println("YP=" + String(YP)  + " XM=" + String(XM));
  Serial.println("YM=" + String(YM)  + " XP=" + String(XP));
}

// Show TFT info Screen
void show_tft(void) {
  tft.setCursor(0, 0);
  tft.setTextSize(2);
  tft.print(F("Found "));
  tft.print(name);
  tft.println(F(" LCD"));
  tft.setTextSize(1);
  tft.print(F("ID=0x"));
  tft.println(identifier, HEX);
  tft.println("Screen is " + String(tft.width()) + "x" + String(tft.height()));
  tft.println("Calibration is: ");
  tft.println("LEFT = " + String(TS_LEFT) + " RT  = " + String(TS_RT));
  tft.println("TOP  = " + String(TS_TOP)  + " BOT = " + String(TS_BOT));
  tft.print("\nWiring is: ");
  if (SwapXY) {
    tft.setTextColor(CYAN);
    tft.setTextSize(2);
  }
  tft.println(SwapXY ? "SWAPXY" : "PORTRAIT");
  tft.println("YP=" + String(YP)  + " XM=" + String(XM));
  tft.println("YM=" + String(YM)  + " XP=" + String(XP));
  tft.setTextSize(2);
  tft.setTextColor(RED);
  tft.setCursor((tft.width() - 48) / 2, (tft.height() * 2) / 4);
  tft.print("EXIT");
  tft.setTextColor(YELLOW, BLACK);
  tft.setCursor(0, (tft.height() * 6) / 8);
  tft.print("Touch screen for loc");
  while (1) {
    tp = ts.getPoint();
    pinMode(XM, OUTPUT);
    pinMode(YP, OUTPUT);
    pinMode(XP, OUTPUT);
    pinMode(YM, OUTPUT);
    if (tp.z < MINPRESSURE || tp.z > MAXPRESSURE) continue;
    if (tp.x > 450 && tp.x < 570  && tp.y > 450 && tp.y < 570) break;
    tft.setCursor(0, (tft.height() * 3) / 4);
    tft.print("tp.x=" + String(tp.x) + " tp.y=" + String(tp.y) + "   ");
  }
}

// Touch Screen Read
void touch_Screen_Read() {
  // Pressure point read
  tp = ts.getPoint();   //tp.x, tp.y are ADC values
  // if sharing pins, you'll need to fix the directions of the touchscreen pins
  pinMode(XM, OUTPUT);
  pinMode(YP, OUTPUT);
  pinMode(XP, OUTPUT);
  pinMode(YM, OUTPUT);
  //    digitalWrite(XM, HIGH);
  //    digitalWrite(YP, HIGH);
  // we have some minimum pressure we consider 'valid'
  // pressure of 0 means no pressing!

  if (tp.z > MINPRESSURE && tp.z < MAXPRESSURE) {
    // is controller wired for Landscape ? or are we oriented in Landscape?
    if (SwapXY != (Orientation & 1)) SWAP(tp.x, tp.y);
    // scale from 0->1023 to tft.width  i.e. left = 0, rt = width
    // most mcufriend have touch (with icons) that extends below the TFT
    // screens without icons need to reserve a space for "erase"
    // scale the ADC values from ts.getPoint() to screen values e.g. 0-239
    if (Orientation == 0) {
      xpos = map(tp.x, TS_RT, TS_LEFT, 0, tft.width());
      ypos = map(tp.y, TS_BOT, TS_TOP, 0, tft.height());


    }
    else if (Orientation == 1) {
      xpos = map(tp.x, TS_LEFT, TS_RT, 0, tft.width());
      ypos = map(tp.y, TS_BOT, TS_TOP, 0, tft.height());
    }
    else if (Orientation == 2) {
      xpos = map(tp.x, TS_RT, TS_LEFT, 0, tft.width());
      ypos = map(tp.y, TS_BOT, TS_TOP, 0, tft.height());
    }
    else if (Orientation == 3) {
      xpos = map(tp.x, TS_RT, TS_LEFT, 0, tft.width());
      ypos = map(tp.y, TS_BOT, TS_TOP, 0, tft.height());
    }
  }
}

// Drawing back button
void drawBackButton() {
  extern const uint8_t BackButton[1024];
  pos_X = 0;
  pos_Y = tft.height() - 35;
  tft.setAddrWindow(pos_X, pos_Y, pos_X + 31, pos_Y + 32);
  tft.pushColors(BackButton, 1024, 1);
}
```

```
// Drawing erase button
void drawEraseButton() {
  extern const uint8_t EraseButton[1024];
  pos_X = tft.width() - 35;
  pos_Y = tft.height() - 35;
  tft.setAddrWindow(pos_X, pos_Y, pos_X + 31, pos_Y + 32);
  tft.pushColors(EraseButton, 1024, 1);
}

// Draw green checkmark
void drawCheckMark(int x, int y) {
  extern const uint8_t CheckMark[256];
  tft.setAddrWindow(x, y, x + 15, y + 16);
  tft.pushColors(CheckMark, 256, 1);
}

// Draw red checkmark
void drawCheckMarkRed(int x, int y) {
  extern const uint8_t CheckMarkRed[256];
  tft.setAddrWindow(x, y, x + 15, y + 16);
  tft.pushColors(CheckMarkRed, 256, 1);
}

// Draw white checkmark
void drawCheckMarkWhite(int x, int y) {
  extern const uint8_t CheckMarkWhite[256];
  tft.setAddrWindow(x, y, x + 15, y + 16);
  tft.pushColors(CheckMarkWhite, 256, 1);
}

// Draw Alarm button X and Y is a position of a button
void drawAlarmButton(int pos_X, int pos_Y) {
  extern const uint8_t AlarmButton[0x1040];
  tft.setAddrWindow(pos_X, pos_Y, pos_X + 64, pos_Y + 65);
  tft.pushColors(AlarmButton, 4160, 1);
}

// Zero all data to load a new screen
void zeroAllData() {
  xpos = -1;
  ypos = -1;
  currentHours = -1;
  currentMinutes = -1;
  currentSeconds = -1;
  temperature = -1;
  currentDate = -1;
  oldDoW = -1;
  PM = -1;
}

// Setup loop for pait. Drawing colors and buttons
void paint_Setup() {
  //show_tft();
  BOXSIZE = tft.width() / 8;
  tft.fillScreen(BLACK);
  tft.fillRect(0, 0, BOXSIZE, BOXSIZE, RED);
  tft.fillRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, YELLOW);
  tft.fillRect(BOXSIZE * 2, 0, BOXSIZE, BOXSIZE, GREEN);
  tft.fillRect(BOXSIZE * 3, 0, BOXSIZE, BOXSIZE, CYAN);
  tft.fillRect(BOXSIZE * 4, 0, BOXSIZE, BOXSIZE, BLUE);
  tft.fillRect(BOXSIZE * 5, 0, BOXSIZE, BOXSIZE, MAGENTA);
  tft.fillRect(BOXSIZE * 6, 0, BOXSIZE, BOXSIZE, ORANGE);
  tft.fillRect(BOXSIZE * 7, 0, BOXSIZE, BOXSIZE, WHITE);
  tft.drawRect(0, 0, BOXSIZE, BOXSIZE, WHITE);
  currentcolor = RED;
  drawBackButton();
  drawEraseButton();
}
```

```
// Paint loop to be called from the main loop when we press on the screen.

void paint_Loop() {

  touch_Screen_Read();

  // are we in top color box area ?

  if (ypos < BOXSIZE) {            //draw white border on selected color box

    oldcolor = currentcolor;


    if (xpos < BOXSIZE) {

      currentcolor = RED;

      tft.drawRect(0, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 2) {

      currentcolor = YELLOW;

      tft.drawRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 3) {

      currentcolor = GREEN;

      tft.drawRect(BOXSIZE * 2, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 4) {

      currentcolor = CYAN;

      tft.drawRect(BOXSIZE * 3, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 5) {

      currentcolor = BLUE;

      tft.drawRect(BOXSIZE * 4, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 6) {

      currentcolor = MAGENTA;

      tft.drawRect(BOXSIZE * 5, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 7) {

      currentcolor = ORANGE;

      tft.drawRect(BOXSIZE * 6, 0, BOXSIZE, BOXSIZE, WHITE);

    } else if (xpos < BOXSIZE * 8) {

      currentcolor = WHITE;

      tft.drawRect(BOXSIZE * 7, 0, BOXSIZE, BOXSIZE, WHITE);

    }
    if (oldcolor != currentcolor) { //rub out the previous white border
      if (oldcolor == RED) tft.fillRect(0, 0, BOXSIZE, BOXSIZE, RED);
      if (oldcolor == YELLOW) tft.fillRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, YELLOW);
      if (oldcolor == GREEN) tft.fillRect(BOXSIZE * 2, 0, BOXSIZE, BOXSIZE, GREEN);
      if (oldcolor == CYAN) tft.fillRect(BOXSIZE * 3, 0, BOXSIZE, BOXSIZE, CYAN);
      if (oldcolor == BLUE) tft.fillRect(BOXSIZE * 4, 0, BOXSIZE, BOXSIZE, BLUE);
      if (oldcolor == MAGENTA) tft.fillRect(BOXSIZE * 5, 0, BOXSIZE, BOXSIZE, MAGENTA);
      if (oldcolor == ORANGE) tft.fillRect(BOXSIZE * 6, 0, BOXSIZE, BOXSIZE, ORANGE);
      if (oldcolor == WHITE) tft.fillRect(BOXSIZE * 7, 0, BOXSIZE, BOXSIZE, WHITE);
    }
  }

  // are we in drawing area ?
  if (((ypos - PENRADIUS) > BOXSIZE) && ((ypos + PENRADIUS) < tft.height())) {
    tft.fillCircle(xpos, ypos, PENRADIUS, currentcolor);
  }

  // are we pressing erase button ?
  if ((ypos > tft.height() - 40) && (xpos > tft.width() - 40) ) {
    // press the bottom of the screen to erase
    tft.fillRect(0, BOXSIZE, tft.width(), tft.height() - BOXSIZE, BLACK);
    xpos = -1;
    ypos = -1;
    drawEraseButton();
    drawBackButton();
  }

  // Are we pressing a back button?
  if ((ypos >= tft.height() - 40) && (xpos <= 40)) {
    xpos = -1;
    ypos = -1;
    currentPage = 0;
    zeroAllData();
    drawHomeScreen();
```

```
 }
}


// Setup of the MCuFriend touchscreen
void TFT_Setup() {


 uint16_t tmp;
 tft.begin(9600);


 tft.reset();
 identifier = tft.readID();


 if (identifier == 0x5310) {
   name = "HX8357D";
   TS_LEFT = 904; TS_RT = 170; TS_TOP = 950; TS_BOT = 158;
   SwapXY = 1;
 }
 else {
   name = "unknown";
 }
 switch (Orientation) {
   case 0:   break;        // No change,  calibrated for PORTRAIT
   case 1: SWAP(TS_LEFT, TS_BOT);  SWAP(TS_TOP, TS_RT); break; // Landscape
   case 2: SWAP(TS_LEFT, TS_RT); SWAP(TS_BOT, TS_TOP); break;
   case 3: SWAP(TS_RT, TS_BOT); SWAP(TS_RT, TS_LEFT); break;
 }


 ts = TouchScreen(XP, YP, XM, YM, 300);     //call the constructor AGAIN with new values.
 tft.begin(identifier);
 show_Serial();
 tft.setRotation(Orientation);
}


// Setting time. To be used once doring setup of rtc. Than comment out this function call in a setup.
```

```
void set_Clock(byte h, byte m, byte s, bool hm) {


  if ((h != "") && (m != "") && (s != "")) {
    rtc.setHour(h);
    rtc.setMinute(m);
    rtc.setSecond(s);
    rtc.setClockMode(hm);
  }
}

// Set date. Same logic.
void set_Date(int mm, int dd, int yr, int doW) {
  if ((mm != "") && (dd != "") && (yr != "") && (doW != "")) {
    rtc.setMonth(mm);
    rtc.setDate(dd);
    rtc.setYear(yr);
    rtc.setDoW(doW);
  }

}

// Drawing a collum X and two Y to draw a collom of any r size and c color
void draw_Column(int x, int y1, int y2, int r, int c) {
  // Draw a top dot divider
  tft.fillCircle(x, y1, r, c);
  // Draw a bottom dot divider
  tft.fillCircle(x, y2, r, c);
}

// Drawing a big clock
void drawHomeClock() {
  // Clock size and color
  tft.setTextSize(10); // Letter size = 65
  tft.setTextColor(GREEN); // Color is green
  pos_X = 50; // Object group beginniing
  pos_Y = 65; // Object group beginniing

  tft.setCursor(pos_X, pos_Y); // Set cursor

  if (currentHours != rtc.getHour(h12, PM)) { // If Hours update

    if (h12 == false) { // If clock in 24 hours format

      if ((rtc.getHour(h12, PM) >= 10)) { // If Hours is a double digit in 24 hours mode
        tft.setCursor(pos_X + 10, pos_Y);
        tft.fillRect(pos_X + 20, pos_Y, pos_X + 65, pos_Y + 20, BLACK);
        currentHours = rtc.getHour(h12, PM); // Get new current time
        tft.print(currentHours); // Print curent hours
      }
      else if ((rtc.getHour(h12, PM) >= 0) && (rtc.getHour(h12,  PM) < 10)) { // If Hours is a single digit in 24
hours mode
        tft.fillRect(pos_X + 10, pos_Y, 105, 70, BLACK);
        tft.setCursor(pos_X + 10, pos_Y);
        tft.print('0');
        tft.setCursor(pos_X + 65, pos_Y);
        currentHours = rtc.getHour(h12,  PM);
        tft.print(currentHours); // Print curent hours
      }
      tft.fillRect(pos_X - 45, pos_Y - 3, 35, 15, BLACK);
      tft.setCursor(pos_X - 45, pos_Y - 3);
      tft.setTextSize(2);
      tft.setTextColor(RED);
      tft.print("24H");
    }

    else if (h12 == true) { // If clock in 12 hours format
      if ((rtc.getHour(h12,  PM) >= 10) && (rtc.getHour(h12,  PM) <= 12)) { // If Hours is a double digit in 24
hours mode
        tft.setCursor(pos_X + 5, pos_Y);
        tft.fillRect(pos_X + 5, pos_Y, pos_X + 60, pos_Y + 5, BLACK);
        currentHours = rtc.getHour(h12,  PM); // Get new current time
        tft.print(currentHours); // Print curent hours
      }

      else if ((rtc.getHour(h12,  PM) >= 0) && (rtc.getHour(h12,  PM) < 10)) { // If Hours is a single digit in
24 hours mode
        tft.fillRect(pos_X + 5, pos_Y, pos_X + 60, pos_Y + 5, BLACK);
        tft.setCursor(pos_X + 5, pos_Y);
        tft.print('0');
        tft.setCursor(pos_X + 65, pos_Y);
        currentHours = rtc.getHour(h12,  PM);
        tft.print(currentHours); // Print curent hours
      }
      tft.setCursor(pos_X - 45, pos_Y);
      tft.setTextSize(3);
      tft.setTextColor(RED);
      if (PM) {
        tft.fillRect(pos_X - 45, pos_Y, 33, 21, BLACK);
        tft.print("PM");
      }
      else {
        tft.fillRect(pos_X - 45, pos_Y, 33, 21, BLACK);
        tft.print("AM");
      }
    }
  }
  // Draw column
  draw_Column(pos_X + 129, pos_Y + 17, pos_Y + 47, 2, GREEN);

  //   Minutes update
```

```
if (currentMinutes != rtc.getMinute()) {
  if ((rtc.getMinute() < 10) && (rtc.getMinute() >= 0)) {
    currentMinutes = rtc.getMinute(); // Getting new minutes
    tft.setTextSize(10);
    tft.setTextColor(GREEN);
    tft.fillRect(pos_X + 145, pos_Y, pos_X + 60, pos_Y + 5, BLACK);
    tft.setCursor(pos_X + 145, pos_Y); // Set cursor
    tft.print('0');
    tft.setCursor(pos_X + 205, pos_Y); // Set cursor
    tft.print(currentMinutes); // Print minutes
  }
  else if (rtc.getMinute() >= 10) {
    currentMinutes = rtc.getMinute(); // Getting new minutes
    tft.setTextSize(10);
    tft.setTextColor(GREEN);
    tft.setCursor(pos_X + 145, pos_Y); // Set cursor
    tft.fillRect(pos_X + 145, pos_Y, pos_X + 60, pos_Y + 5, BLACK);
    tft.print(currentMinutes); // Print minutes
  }
}
// Draw column
draw_Column(pos_X + 270, pos_Y + 17, pos_Y + 47, 2, GREEN);

// Draw seconds
if (currentSeconds != rtc.getSecond()) {
  if ((rtc.getSecond() >= 0) && (rtc.getSecond() < 10)) {
    currentSeconds = rtc.getSecond(); // Getting new Seconds
    tft.fillRect(pos_X + 285, pos_Y, pos_X + 60, pos_Y + 5, BLACK);
    tft.setCursor(pos_X + 285, pos_Y);
    tft.print('0');
    tft.setCursor(pos_X + 285 + 60, pos_Y); // Set cursor
    tft.print(currentSeconds); // Print Seconds
  }
  else {
      currentSeconds = rtc.getSecond(); // Getting new Seconds
      tft.setCursor(pos_X + 285, pos_Y); // Set cursor
      tft.fillRect(pos_X + 285, pos_Y, pos_X + 60, pos_Y + 5, BLACK);
      tft.print(currentSeconds); // Print Seconds
    }
  }
}

// Drwaing a small clock
void drawSmallClock() {

  // Setting up coordinates for a small clock begginning
  int  pos_X_SC = 200; // Object group beginniing
  int  pos_Y_SC = 7; // Object group beginniing
  int text_Size = 2;
  // Clock size and color
  tft.setTextColor(GREEN); // Color is green
  tft.setCursor(pos_X_SC, pos_Y_SC); // Set cursor
  tft.setTextSize(text_Size); // Size of the didgits
  if (currentHours != rtc.getHour(h12, PM)) { // If Hours update
    if (h12 == false) { // If clock in 24 hours format
      if ((rtc.getHour(h12, PM) >= 10)) { // If Hours is a double digit in 24 hours mode
        tft.setCursor(pos_X_SC + 10, pos_Y_SC);
        tft.fillRect(pos_X_SC + 10, pos_Y_SC, 22, 14, BLACK);
        currentHours = rtc.getHour(h12, PM); // Get new current time
        tft.print(currentHours); // Print curent hours
      }
      else if ((rtc.getHour(h12, PM) >= 0) && (rtc.getHour(h12,  PM) < 10)) { // If Hours is a single digit in 24
hours mode
        tft.setCursor(pos_X_SC + 10, pos_Y_SC);
        tft.print('0');
        tft.setCursor(pos_X_SC + 22, pos_Y_SC);
        tft.fillRect(pos_X_SC + 22, pos_Y_SC, 10, 14, BLACK);
        currentHours = rtc.getHour(h12,  PM);
      tft.print(currentHours); // Print curent hours

     }

     // Print 24 hours icon

     tft.fillRect(pos_X_SC - 30, pos_Y_SC, 35, 14, BLACK);

     tft.setCursor(pos_X_SC - 30, pos_Y_SC);

     tft.setTextSize(text_Size);

     tft.setTextColor(RED);

     tft.print("24H");

    }


    else if (h12 == true) { // If clock in 12 hours format


     if ((rtc.getHour(h12,  PM) >= 10) && (rtc.getHour(h12,  PM) <= 12)) { // If Hours is a double digit in 24
hours mode

      tft.setCursor(pos_X_SC + 10, pos_Y_SC);

      tft.fillRect(pos_X_SC + 10, pos_Y_SC, 22, 14, BLACK);

      currentHours = rtc.getHour(h12, PM); // Get new current time

      tft.print(currentHours); // Print curent hours

     }

     else if ((rtc.getHour(h12,  PM) >= 0) && (rtc.getHour(h12,  PM) < 10)) { // If Hours is a single digit in
24 hours mode

      tft.fillRect(pos_X_SC + 10, pos_Y_SC, 22, 14, BLACK);

      tft.setCursor(pos_X_SC + 10, pos_Y_SC);

      tft.print('0');

      tft.setCursor(pos_X_SC + 22, pos_Y_SC);

      currentHours = rtc.getHour(h12,  PM);

      tft.print(currentHours); // Print curent hours


     }

     tft.setCursor(pos_X_SC - 20, pos_Y_SC);

     tft.setTextSize(text_Size);

     tft.setTextColor(RED);

     if (PM) {

      tft.fillRect(pos_X_SC - 20, pos_Y_SC, 23, 16, BLACK);
      tft.print("PM");
    }
    else {
      tft.fillRect(pos_X_SC - 20, pos_Y_SC, 23, 16, BLACK);
      tft.print("AM");
    }
  }
}

// Draw column
draw_Column(pos_X_SC + 37, pos_Y_SC + 3, pos_Y_SC + 10, 1, GREEN);

// Returning size and color to the value of 2 and green again
tft.setTextSize(text_Size); // Size is 1
tft.setTextColor(GREEN); // Color is green

//   Minutes update
if (currentMinutes != rtc.getMinute()) {

  if ((rtc.getMinute() < 10) && (rtc.getMinute() >= 0)) {
    currentMinutes = rtc.getMinute(); // Getting new minutes
    tft.setTextColor(GREEN);
    tft.fillRect(pos_X_SC + 43, pos_Y_SC, 22, 14, BLACK);
    tft.setCursor(pos_X_SC + 43, pos_Y_SC); // Set cursor
    tft.print('0');
    tft.setCursor(pos_X_SC + 55, pos_Y_SC); // Set cursor
    tft.print(currentMinutes); // Print minutes
  }
  else if (rtc.getMinute() >= 10) {
    currentMinutes = rtc.getMinute(); // Getting new minutes
    tft.setCursor(pos_X_SC + 43, pos_Y_SC); // Set cursor
    tft.fillRect(pos_X_SC + 43, pos_Y_SC, 22, 14, BLACK);
    tft.print(currentMinutes); // Print minutes
  }
```

```
 }

 // Draw column
 draw_Column(pos_X_SC + 70, pos_Y_SC + 3, pos_Y_SC + 10, 1, GREEN);

 // Draw seconds
 if (currentSeconds != rtc.getSecond()) {
  if ((rtc.getSecond() >= 0) && (rtc.getSecond() < 10)) {
   currentSeconds = rtc.getSecond(); // Getting new Seconds
   tft.fillRect(pos_X_SC + 76, pos_Y_SC, 22, 14, BLACK);
   tft.setCursor(pos_X_SC + 76, pos_Y_SC);
   tft.print('0');
   tft.setCursor(pos_X_SC + 88, pos_Y_SC); // Set cursor
   tft.print(currentSeconds); // Print Seconds
  }
  else {
   currentSeconds = rtc.getSecond(); // Getting new Seconds
   tft.setCursor(pos_X_SC + 76, pos_Y_SC); // Set cursor
   tft.fillRect(pos_X_SC + 76, pos_Y_SC, 22, 14, BLACK);
   tft.print(currentSeconds); // Print Seconds
  }
 }
}

// Drawing media/Play button. We are short on SRAM memory because of all of the Icons in PROGMEM
void drawMediaButton(int X, int Y) {
 extern const uint8_t MusicPlayerButton[0x1040]; // Declaring external arraey
 pos_X = X; // Set position
 pos_Y = Y;
 tft.setAddrWindow(pos_X, pos_Y, pos_X + 64, pos_Y + 65); // Draw Object window
 tft.pushColors(MusicPlayerButton, 4160, 1);
}

// Draw Paint button
```

```
void drawPaintButton() {
  extern const uint8_t PaintButton[4225];
  pos_X = 207;
  pos_Y = 170;
  tft.setAddrWindow(pos_X, pos_Y, pos_X + 64, pos_Y + 65);
  tft.pushColors(PaintButton, 4224, 1);
}

void drawTemp() {
  // Print temperature in a left top corner
  if (temperature != rtc.getTemperature()) {
    pos_X = 7;
    pos_Y = 6;
    temperature = rtc.getTemperature();
    tft.fillRect(pos_X + 25, pos_Y, pos_X + 15, pos_Y + 9, BLACK);
    tft.fillRect(pos_X + 61, pos_Y, pos_X + 15, pos_Y + 9, BLACK);
    tft.setTextColor(WHITE); // Sets color to white
    tft.setTextSize(2); // Sets font to big
    tft.setCursor(pos_X, 7);
    tft.print("T:");
    tft.setCursor(pos_X + 25, 7);
    tft.print(rtc.getTemperature());
    tft.setTextSize(1);
    tft.setCursor(pos_X + 85, 3);
    tft.print('o');
    tft.setTextSize(2);
    tft.setCursor(pos_X + 94, 7);
    tft.print("C");
  }
}

// Draw Temp
void drawDate() {
  // Print date
```

```
if (currentDate != rtc.getDate()) {
 currentDate = rtc.getDate();
 tft.setTextColor(WHITE); // Sets color to white
 tft.setTextSize(2); // Sets font to big
 int     pos_X_Date = tft.width() - 107;
 int    pos_Y_Date = 7;
 tft.fillRect(pos_X_Date, pos_Y_Date, 102, 14, BLACK);
 if (rtc.getMonth(Century) < 10) tft.setCursor(pos_X_Date + 10, pos_Y_Date);
 else tft.setCursor(pos_X_Date - 2, pos_Y_Date);
 tft.print(rtc.getMonth(Century), DEC);
 tft.setCursor(pos_X_Date + 18 , pos_Y_Date);
 tft.print(".");

 if (rtc.getDate() < 10) {
   tft.setCursor(pos_X_Date + 28, pos_Y_Date);
   tft.print(0);
   tft.setCursor(pos_X_Date + 40, pos_Y_Date);
   tft.print(rtc.getDate(), DEC);
 }
 else if (rtc.getDate() >= 10) {
   tft.setCursor(pos_X_Date + 28, pos_Y_Date);
   tft.print(rtc.getDate(), DEC);
 }

 tft.setCursor(pos_X_Date + 48 , pos_Y_Date);
 tft.print('.');
 tft.setCursor(pos_X_Date + 58, pos_Y_Date);
 tft.print("2");
 tft.setCursor(pos_X_Date + 70, pos_Y_Date);
 if (Century == false) {
   tft.print('0');
 } else tft.print('1');
 tft.setCursor(pos_X_Date + 80, pos_Y_Date);
 tft.print(rtc.getYear());
```

```
  }
}

// Get Day of the week from RTC
void dow() {
  DoW = rtc.getDoW(); //  Get new day of the week
  // Day of the week switch case
  switch (DoW) {
    case 1:
      day_Of_The_Week = "Sunday";
      break;
    case 2:
      day_Of_The_Week = "Monday";
      break;
    case 3:
      day_Of_The_Week = "Tuesday";
      break;
    case 4:
      day_Of_The_Week = "Wednesday";
      break;
    case 5:
      day_Of_The_Week = "Thursday";
      break;
    case 6:
      day_Of_The_Week = "Friday";
      break;
    case 7:
      day_Of_The_Week = "Saturday";
      break;
  }
}

// Drow new day of the week if any change accuers doring the cycle
void drawDayOfTheWeek() {
```

```
// Print day of the week
if (oldDoW != DoW) {
 oldDoW = DoW;
 tft.fillRect(369, 23, 106, 17, BLACK);
 tft.setTextSize(2); // Sets font to big
 pos_X = tft.width() - 107;
 if (DoW == 1) {
  tft.setTextColor(GREEN);
  tft.setCursor(pos_X + 32, 24);
  tft.print(day_Of_The_Week);
 }
 if (DoW == 2) {
  tft.setTextColor(YELLOW);
  tft.setCursor(pos_X + 32, 24);
  tft.print(day_Of_The_Week);
 }
 if (DoW == 3) {
  tft.setTextColor(YELLOW);
  tft.setCursor(pos_X + 20, 24);
  tft.print(day_Of_The_Week);
 }
 if (DoW == 4) {
  tft.setTextColor(YELLOW);
  tft.setCursor(pos_X - 4, 24);
  tft.print(day_Of_The_Week);
 }
 if (DoW == 5) {
  tft.setTextColor(YELLOW);
  tft.setCursor(pos_X + 8, 24);
  tft.print(day_Of_The_Week);
 }
 if (DoW == 6) {
  tft.setTextColor(YELLOW);
  tft.setCursor(pos_X + 32, 24);
```

```
      tft.print(day_Of_The_Week);
    }
    if (DoW == 7) {
      tft.setTextColor(GREEN);
      tft.setCursor(pos_X + 8, 24);
      tft.print(day_Of_The_Week);
    }


  }
}
```

// Draw alarm Status. Is unused function. Im Storing my name here. But in a future I will add a string from the new alarm data set.

```
void drawAlarmStatus() {
  // Check if alarm is ON or OFF
  if (alarmString == "") {
    tft.setTextSize(2);
    tft.setTextColor(GREEN);
    tft.setCursor ((tft.width() / 2) - 90, 299);
    tft.print("by Roman Yushchyk");
  }
  else {
    tft.setTextColor(GREEN);
    tft.setTextSize(2);
    tft.setCursor((tft.width() / 2) - 75, 280);
    tft.print("Alarm set for: ");
    tft.setCursor((tft.width() / 2) - 75 , 299);
    tft.print(alarmString);
  }
}


// File from SD tough SDfat 16-bit file sysyetm
uint16_t read16(File& f) {
  uint16_t result;        // read little-endian
  f.read(&result, sizeof(result));
  return result;
}

// File from SD tough SDfat 32-bit file sysyetm
uint32_t read32(File& f) {
  uint32_t result;
  f.read(&result, sizeof(result));
  return result;
}

// Draw bitmam file. Image
uint8_t showBMP(char *nm, int x, int y) {
  File bmpFile;
  int bmpWidth, bmpHeight;    // W+H in pixels
  uint8_t bmpDepth;           // Bit depth (currently must be 24, 16, 8, 4, 1)
  uint32_t bmpImageoffset;    // Start of image data in file
  uint32_t rowSize;           // Not always = bmpWidth; may have padding
  uint8_t sdbuffer[3 * BUFFPIXEL];    // pixel in buffer (R+G+B per pixel)
  uint16_t lcdbuffer[(1 << PALETTEDEPTH) + BUFFPIXEL], *palette = NULL;
  uint8_t bitmask, bitshift;
  boolean flip = true;        // BMP is stored bottom-to-top
  int w, h, row, col, lcdbufsiz = (1 << PALETTEDEPTH) + BUFFPIXEL, buffidx;
  uint32_t pos;               // seek position
  boolean is565 = false;      //

  uint16_t bmpID;
  uint16_t n;                 // blocks read
  uint8_t ret;

  if ((x >= tft.width()) || (y >= tft.height()))
    return 1;               // off screen

  bmpFile = SD.open(nm);      // Parse BMP header
```

```
bmpID = read16(bmpFile);    // BMP signature
(void) read32(bmpFile);     // Read & ignore file size
(void) read32(bmpFile);     // Read & ignore creator bytes
bmpImageoffset = read32(bmpFile);       // Start of image data
(void) read32(bmpFile);     // Read & ignore DIB header size
bmpWidth = read32(bmpFile);
bmpHeight = read32(bmpFile);
n = read16(bmpFile);        // # planes -- must be '1'
bmpDepth = read16(bmpFile); // bits per pixel
pos = read32(bmpFile);      // format
if (bmpID != 0x4D42) ret = 2; // bad ID
else if (n != 1) ret = 3;   // too many planes
else if (pos != 0 && pos != 3) ret = 4; // format: 0 = uncompressed, 3 = 565
else if (bmpDepth < 16 && bmpDepth > PALETTEDEPTH) ret = 5; // palette
else {
  bool first = true;
  is565 = (pos == 3);               // ?already in 16-bit format
  // BMP rows are padded (if needed) to 4-byte boundary
  rowSize = (bmpWidth * bmpDepth / 8 + 3) & ~3;
  if (bmpHeight < 0) {              // If negative, image is in top-down order.
    bmpHeight = -bmpHeight;
    flip = false;
  }

  w = bmpWidth;
  h = bmpHeight;
  if ((x + w) >= tft.width())       // Crop area to be loaded
    w = tft.width() - x;
  if ((y + h) >= tft.height())      //
    h = tft.height() - y;

  if (bmpDepth <= PALETTEDEPTH) {   // these modes have separate palette
    bmpFile.seek(BMPIMAGEOFFSET); //palette is always @ 54
    bitmask = 0xFF;
  if (bmpDepth < 8)
    bitmask >>= bmpDepth;
  bitshift = 8 - bmpDepth;
  n = 1 << bmpDepth;
  lcdbufsiz -= n;
  palette = lcdbuffer + lcdbufsiz;
  for (col = 0; col < n; col++) {
    pos = read32(bmpFile);    //map palette to 5-6-5
    palette[col] = ((pos & 0x0000F8) >> 3) | ((pos & 0x00FC00) >> 5) | ((pos & 0xF80000) >> 8);
  }
}

// Set TFT address window to clipped image bounds







tft.setAddrWindow(x, y, x + w - 1, y + h - 1);
for (row = 0; row < h; row++) { // For each scanline...
  // Seek to start of scan line.  It might seem labor-
  // intensive to be doing this on every line, but this
  // method covers a lot of gritty details like cropping
  // and scanline padding.  Also, the seek only takes
  // place if the file position actually needs to change
  // (avoids a lot of cluster math in SD library).
  uint8_t r, g, b, *sdptr;
  int lcdidx, lcdleft;
  if (flip)   // Bitmap is stored bottom-to-top order (normal BMP)
    pos = bmpImageoffset + (bmpHeight - 1 - row) * rowSize;
  else        // Bitmap is stored top-to-bottom
    pos = bmpImageoffset + row * rowSize;
  if (bmpFile.position() != pos) { // Need seek?
    bmpFile.seek(pos);
  buffidx = sizeof(sdbuffer); // Force buffer reload
}

for (col = 0; col < w; ) {  //pixels in row
  lcdleft = w - col;
  if (lcdleft > lcdbufsiz) lcdleft = lcdbufsiz;
  for (lcdidx = 0; lcdidx < lcdleft; lcdidx++) { // buffer at a time
    uint16_t color;
    // Time to read more pixel data?
    if (buffidx >= sizeof(sdbuffer)) { // Indeed
      bmpFile.read(sdbuffer, sizeof(sdbuffer));
      buffidx = 0; // Set index to beginning
      r = 0;
    }
    switch (bmpDepth) {          // Convert pixel from BMP to TFT format
      case 24:
        b = sdbuffer[buffidx++];
        g = sdbuffer[buffidx++];
        r = sdbuffer[buffidx++];
        color = tft.color565(r, g, b);
        break;
      case 16:
        b = sdbuffer[buffidx++];
        r = sdbuffer[buffidx++];
        if (is565)
          color = (r << 8) | (b);
        else
          color = (r << 9) | ((b & 0xE0) << 1) | (b & 0x1F);
        break;
      case 1:
      case 4:
      case 8:
        if (r == 0)
          b = sdbuffer[buffidx++], r = 8;
```

```
            color = palette[(b >> bitshift) & bitmask];
            r -= bmpDepth;
            b <<= bmpDepth;
            break;
          }
          lcdbuffer[lcdidx] = color;

        }
        tft.pushColors(lcdbuffer, lcdidx, first);
        first = false;
        col += lcdidx;
      }          // end cols
    }              // end rows
    tft.setAddrWindow(0, 0, tft.width() - 1, tft.height() - 1); //restore full screen
    ret = 0;       // good render
  }
  bmpFile.close();
  return (ret);
}

// Draw Background loop. Used only once for the main screen.
void drawBackgroundLoop() {

  char *nm = namebuf + pathlen;
  //root.rewindDirectory(); // To display only the first image
  File f = root.openNextFile();
  uint8_t ret;
  uint32_t start;
  if (f != NULL) {
#ifdef USE_SDFAT
    f.getName(nm, 32 - pathlen);
#else
    strcpy(nm, (char *)f.name());
#endif
```

```
f.close();
strlwr(nm);
if (strstr(nm, ".bmp") != NULL && strstr(nm, NAMEMATCH) != NULL) {
  Serial.print(namebuf);
  Serial.print(F(" - "));
  tft.fillScreen(0);
  start = millis();
  ret = showBMP(namebuf, 0, 0);
  switch (ret) {
    case 0:
      Serial.print(millis() - start);
      Serial.println(F("ms"));
      delay(5000);
      break;
    case 1:
      Serial.println(F("bad position"));
      break;
    case 2:
      Serial.println(F("bad BMP ID"));
      break;
    case 3:
      Serial.println(F("wrong number of planes"));
      break;
    case 4:
      Serial.println(F("unsupported BMP format"));
      break;
    case 5:
      Serial.println(F("unsupported palette"));
      break;
    default:
      Serial.println(F("unknown"));
      break;
  }
}
```

```
  }
  else root.rewindDirectory();
}


// Draw Main screen
void drawHomeScreen() {


  dow();   // Get new day of the week
  //tft.fillScreen(BLACK);                    ///////////////////////////////Just Black
background///////////////////////
  drawBackgroundLoop();                    //////////////////////////////////////// BMP
//////////////////////////////


  zeroAllData();
  drawAlarmStatus();
  drawDayOfTheWeek(); // Draw day of the week


  drawTemp();
  drawDate();
  drawHomeClock();
  drawAlarmButton(365, 170);
  drawMediaButton(50, 170);
  drawPaintButton();
}


// Draw Radio button
void drawRadioButton() {
  extern const uint8_t RadioButton[4225];
  int pos_XRB = 207;
  int pos_YRB = 140;
  tft.setAddrWindow(pos_XRB, pos_YRB, pos_XRB + 64, pos_YRB + 65);
  tft.pushColors(RadioButton, 4225, 1);
}
```

```
// Draw Bloetoth button
void drawBluetoothButton() {
  extern const uint8_t BluetoothButton[825];
  int pos_XBB = 345;
  int pos_YBB = 160;
  tft.setAddrWindow(pos_XBB, pos_YBB, pos_XBB + 24, pos_YBB + 33);
  tft.pushColors(BluetoothButton, 816, 1);
}

// Draw pause button
void drawPause(int X, int Y) {
  extern const uint8_t ButtonPause[3600];
  int  pos_XPAUSE = X;
  int  pos_YPAUSE = Y;
  tft.setAddrWindow(pos_XPAUSE, pos_YPAUSE, pos_XPAUSE + 59, pos_YPAUSE + 60);
  tft.pushColors(ButtonPause, 3600, 1);
}

// Draw Media Button
void draw_Media_Screen() {
  tft.fillScreen(BLACK);
  drawRadioButton();
  drawMediaButton(87, 140);
  drawBluetoothButton();
  drawBackButton();
}

// Draw previus Button
void drawPreviousButton() {
  extern const uint8_t PreviousButton[0x9C4];
  int  pos_XPB = 95;
  int  pos_YPB = 135;
  tft.setAddrWindow(pos_XPB, pos_YPB, pos_XPB + 49, pos_YPB + 50);
  tft.pushColors(PreviousButton, 2496, 1);
```

```
}

// Draw Next Button for the music player
void drawNextButton() {
  extern const uint8_t NextButton[0x9C4];
  int  pos_XNB = 335;
  int  pos_YNB = 135;
  tft.setAddrWindow(pos_XNB, pos_YNB, pos_XNB + 49, pos_YNB + 50);
  tft.pushColors(NextButton, 2496, 1);

}

// Draw Volume Down Button
void drawVolumeDown() {
  extern const uint8_t VolumeDown[0x170];
  int  pos_XVD = 25;
  int  pos_YVD = 148;
  tft.setAddrWindow(pos_XVD, pos_YVD, pos_XVD + 15, pos_YVD + 23);
  tft.pushColors(VolumeDown, 368, 1);
}

// Draw Volume Up Button
void drawVolumeUp() {
  extern const uint8_t VolumeUp[0x3B8];
  int  pos_XVU = tft.width() - 58;
  int  pos_YVU = 148;
  tft.setAddrWindow(pos_XVU, pos_YVU, pos_XVU + 33, pos_YVU + 28);
  tft.pushColors(VolumeUp, 944, 1);
}

// Draw Mp3 Player screen
void mp3_Player_Screen() {
  tft.fillScreen(BLACK);
  drawPreviousButton();
  drawNextButton();

  drawBackButton();

  drawVolumeDown();

  drawVolumeUp();

}


// Draw Radio Screen. Under Development.

void draw_Radio_Screen() {

  tft.fillScreen(BLACK);

  drawBackButton();

}


// Check if alarm is enabled

void checkAlarmStatus(int n) {

  if (rtc.checkAlarmEnabled(n)) {

    tft.setTextColor(GREEN);

    tft.println("On");

  }

  else {

    tft.setTextColor(RED);

    tft.println("Off");

  }

}


// Store Alarm One to EEPROM memory. You can rindomize it the setup loop and declaration section if
you uncoment ramdom() and seed() attributus

void storeAlarmOneToEEPROM() {

  for (int i = 0; i <= 6; i++) {

    Serial.print("Alarm One EEPROM #");

    Serial.print(i);

    Serial.print(" is set for: ");

    Serial.println(alarmOneWeek[i]);

    EEPROM.write(eeAddressAlarmOne + i, alarmOneWeek[i]);

  }
```

```
}


// Store alarm Two to the EEPROM

void storeAlarmTwoToEEPROM() {

  for (int i = 0; i <= 6; i++) {

    Serial.print("Alarm Two EEPROM #");

    Serial.print(i);

    Serial.print(" is set for: ");

    Serial.println(alarmTwoWeek[i]);

    EEPROM.write(eeAddressAlarmTwo + i, alarmTwoWeek[i]);

  }

}


// Set alarm 1 or 2 int a. Detail instructions in a setup

void setAlarm(int a, byte ADay, byte AHour, byte AMinute, byte ASeconds, byte AlarmBits, bool ADy,
bool Ah12, bool APM) {

  if (a == 1) {

    rtc.setA1Time(ADay, AHour, AMinute, AlarmBits, ASeconds, ADy, Ah12, APM); // dOfW_Date True for
days of the week false for a date

    if (ADy)storeAlarmOneToEEPROM();

  }

  else if (a == 2) {

    rtc.setA2Time(ADay, AHour, AMinute, AlarmBits, ADy, Ah12, APM); // dOfW_Date True for days of the
week false for a date

    if (ADy)storeAlarmTwoToEEPROM();

  }

}


// Get alarm. Both

void getAlarm(byte& A1Day, byte& A1Hour, byte& A1Minute, byte& A1Second, byte& A1Bits, bool&
A1Dy, bool& A1h12, bool& A1PM, byte& A2Day, byte& A2Hour, byte& A2Minute, byte& A2Bits, bool&
A2Dy, bool& A2h12, bool& A2PM) {

  rtc.getA1Time(A1Day, A1Hour, A1Minute, A1Second, A1Bits, A1Dy, A1h12, A1PM);

  rtc.getA2Time(A2Day, A2Hour, A2Minute, A2Bits, A2Dy, A2h12, A2PM);
```

```
}


// Retrive weeks arreys from EEPROM
void getAlarmWeeksFromEEPROM() {
 for (int i = 0; i <= 6; i++) {
   Serial.print("Alarm One get from EEPROM #");
   Serial.print(i);
   Serial.print(" is set for: ");
   EEPROM.get(eeAddressAlarmOne + i, alarmOneWeek[i]);
   Serial.println(alarmOneWeek[i]);
 }
 for (int i = 0; i <= 6; i++) {
   Serial.print("Alarm Two get from EEPROM #");
   Serial.print(i);
   Serial.print(" is set for: ");
   EEPROM.get(eeAddressAlarmTwo + i, alarmTwoWeek[i]);
   Serial.println(alarmTwoWeek[i]);
 }
}


// Draw Alarm Screen
void draw_Alarm_Screen() {
 drawSmallClock(); // Initiate clock
 drawBackButton(); // Draw back button


 // draw alarm 1
 drawAlarmButton(X_A1, Y_A1 - 30);
 tft.setCursor(X_A1 - 19, Y_A1 + 35);
 tft.setTextColor(GreenYellow);
 tft.print("Alarm One");
 tft.setCursor(X_A1 - 19, Y_A1 + 51);
 tft.print("Status:");
 checkAlarmStatus(1);
 tft.setCursor(X_A1 - 19, Y_A1 + 67);
 tft.setTextColor(GreenYellow);
 tft.print("Set for:");
 tft.setCursor(X_A1 - 49, Y_A1 + 93);
 tft.setTextColor(RED);

 if (A1h12) {
   if (A1PM) {
     tft.print("PM");
     tft.setCursor(X_A1 - 11, Y_A1 + 94);
     // Set text size
     tft.setTextSize(4);
     // Set color to blue
     tft.setTextColor(BLUE);
     // Print alarm 1 Hour
     if (A1Hour < 10) {
       tft.print('0');
       tft.setCursor(X_A1 + 11, Y_A1 + 94);
       tft.print(A1Hour);
     } else tft.print(A1Hour);
     // Draw column
     draw_Column(X_A1 + 36, Y_A1 + 97, Y_A1 + 118, 1, GREEN);
     // Draw alarm 1 minutes
     tft.setCursor(X_A1 + 43, Y_A1 + 94);
     if (A1Minute < 10) {
       tft.print('0');
       tft.setCursor(X_A1 + 67, Y_A1 + 94);
       tft.print(A1Minute);
     } else tft.print(A1Minute);
   }
   else {
     tft.print("AM");
     tft.setCursor(X_A1 - 11, Y_A1 + 94);
     // Set text size
     tft.setTextSize(4);
     // Set color to blue
     tft.setTextColor(BLUE);
     // Print Hour
     if (A1Hour < 10) {
       tft.print('0');
       tft.setCursor(X_A1 + 11, Y_A1 + 94);
       tft.print(A1Hour);
     } else tft.print(A1Hour);      // Draw column
     draw_Column(X_A1 + 36, Y_A1 + 97, Y_A1 + 118, 1, GREEN);
     // Draw alarm 1 minutes
     tft.setCursor(X_A1 + 43, Y_A1 + 94);
     if (A1Minute < 10) {
       tft.print('0');
       tft.setCursor(X_A1 + 67, Y_A1 + 94);
       tft.print(A1Minute);
     } else tft.print(A1Minute);
   }
 }
 else { // 24 hours format
   tft.print("24H");
   tft.setCursor(X_A1, Y_A1 + 94);
   // Set text size
   tft.setTextSize(4);
   // Set color to blue
   tft.setTextColor(BLUE);
   // Print Hour
   if (newA1Hour < 10) {
     tft.print('0');
     tft.setCursor(X_A1 + 24, Y_A1 + 94);
     tft.print(A1Hour);
   } else tft.print(A1Hour);
   // Draw column
   draw_Column(X_A1 + 49, Y_A1 + 97, Y_A1 + 118, 1, GREEN);
   // Draw alarm 1 minutes
   tft.setCursor(X_A1 + 55, Y_A1 + 94);
   if (A1Minute < 10) {
     tft.print('0');
     tft.setCursor(X_A1 + 79, Y_A1 + 94);
     tft.print(A1Minute);
   } else tft.print(A1Minute);
 }

 // Draw day check boxses
 tft.drawRect(X_A1 - 41, Y_A1 + 132, 20, 20, WHITE);
 tft.drawRect(X_A1 - 19, Y_A1 + 132, 20, 20, WHITE);
 tft.drawRect(X_A1 + 3, Y_A1 + 132, 20, 20, WHITE);
 tft.drawRect(X_A1 + 25, Y_A1 + 132, 20, 20, WHITE);
 tft.drawRect(X_A1 + 47, Y_A1 + 132, 20, 20, WHITE);
 tft.drawRect(X_A1 + 69, Y_A1 + 132, 20, 20, WHITE);
 tft.drawRect(X_A1 + 91, Y_A1 + 132, 20, 20, WHITE);

 // Draw Set and Clear buttons
 tft.drawRect(X_A1 - 19, Y_A1 + 182, 50, 25, WHITE);
 tft.setTextSize(2);
 tft.setTextColor(GREEN);
 tft.setCursor(X_A1 - 11, Y_A1 + 188);
 tft.print("SET");
 tft.drawRect(X_A1 + 39, Y_A1 + 182, 50, 25, WHITE);
 tft.setTextColor(ORANGE);
 tft.setCursor(X_A1 + 47, Y_A1 + 188);
 tft.print("Clr");

 // Draw date check box
 tft.drawRect(X_A1 + 11, Y_A1 + 157, 20, 20, WHITE);

 // Draw current settings date
 tft.drawRect(X_A1 + 39, Y_A1 + 157, 50, 20, WHITE);
```

```
// Draw plus and minus sighn
tft.drawRect(X_A1 + 93, Y_A1 + 182, 60, 20, RED);
tft.setCursor(X_A1 + 98, Y_A1 + 182);
tft.setTextSize(3);
tft.setTextColor(RED);
tft.print(" + ");
tft.drawRect(X_A1 + 93, Y_A1 + 157, 60, 20, BLUE);
tft.setCursor(X_A1 + 98, Y_A1 + 156);
tft.setTextColor(BLUE);
tft.print(" - ");

// Return size and color back to default
tft.setTextSize(2);
tft.setTextColor(GREEN);
tft.setCursor(X_A1 - 11, Y_A1 + 161);



// draw alarm 2
drawAlarmButton(X_A2, Y_A2 - 30);
tft.setCursor(X_A2 - 19, Y_A2 + 35);
tft.setTextColor(GreenYellow);
tft.print("Alarm Two");
tft.setCursor(X_A2 - 19, Y_A2 + 51);
tft.print("Status:");
checkAlarmStatus(2);
tft.setCursor(X_A2 - 19, Y_A2 + 67);
tft.setTextColor(GreenYellow);
tft.print("Set for:");
tft.setCursor(X_A2 - 49, Y_A2 + 93);
tft.setTextColor(RED);

if (A2h12) {
  if (A2PM) {
    tft.print("PM");
    tft.setCursor(X_A2 - 11, Y_A2 + 94);
    // Set text size
    tft.setTextSize(4);
    // Set color to blue
    tft.setTextColor(BLUE);
    // Print alarm 1 Hour
    if (A2Hour < 10) {
      tft.print('0');
      tft.setCursor(X_A2 + 11, Y_A2 + 94);
      tft.print(A2Hour);
    } else tft.print(A2Hour);
    // Draw column
    draw_Column(X_A2 + 36, Y_A2 + 97, Y_A2 + 118, 1, GREEN);
    // Draw alarm 1 minutes
    tft.setCursor(X_A2 + 43, Y_A2 + 94);
    if (A2Minute < 10) {
      tft.print('0');
      tft.setCursor(X_A2 + 67, Y_A2 + 94);
      tft.print(A2Minute);
    } else tft.print(A2Minute);
  }
  else {
    tft.print("AM");
    tft.setCursor(X_A2 - 11, Y_A2 + 94);
    // Set text size
    tft.setTextSize(4);
    // Set color to blue
    tft.setTextColor(BLUE);
    // Print Hour
    if (A2Hour < 10) {
      tft.print('0');
      tft.setCursor(X_A2 + 11, Y_A2 + 94);
      tft.print(A2Hour);
    } else tft.print(A2Hour);     // Draw column
```

```
    draw_Column(X_A2 + 36, Y_A2 + 97, Y_A2 + 118, 1, GREEN);
    // Draw alarm 1 minutes
    tft.setCursor(X_A2 + 43, Y_A2 + 94);
    if (A2Minute < 10) {
      tft.print('0');
      tft.setCursor(X_A2 + 67, Y_A2 + 94);
      tft.print(A2Minute);
    } else tft.print(A2Minute);
  }
}
else { // 24 hours format
  tft.print("24H");
  tft.setCursor(X_A2, Y_A2 + 94);
  // Set text size
  tft.setTextSize(4);
  // Set color to blue
  tft.setTextColor(BLUE);
  // Print Hour
  if (newA2Hour < 10) {
    tft.print('0');
    tft.setCursor(X_A2 + 24, Y_A2 + 94);
    tft.print(A2Hour);
  } else tft.print(A2Hour);
  // Draw column
  draw_Column(X_A2 + 49, Y_A2 + 97, Y_A2 + 118, 1, GREEN);
  // Draw alarm 1 minutes
  tft.setCursor(X_A2 + 55, Y_A2 + 94);
  if (A2Minute < 10) {
    tft.print('0');
    tft.setCursor(X_A2 + 79, Y_A2 + 94);
    tft.print(A2Minute);
  } else tft.print(A2Minute);
}
```

```
tft.drawRect(X_A2 - 41, Y_A2 + 132, 20, 20, WHITE);
tft.drawRect(X_A2 - 19, Y_A2 + 132, 20, 20, WHITE);
tft.drawRect(X_A2 + 3, Y_A2 + 132, 20, 20, WHITE);
tft.drawRect(X_A2 + 25, Y_A2 + 132, 20, 20, WHITE);
tft.drawRect(X_A2 + 47, Y_A2 + 132, 20, 20, WHITE);
tft.drawRect(X_A2 + 69, Y_A2 + 132, 20, 20, WHITE);
tft.drawRect(X_A2 + 91, Y_A2 + 132, 20, 20, WHITE);

// Draw Set and Clear buttons
tft.drawRect(X_A2 - 19, Y_A2 + 182, 50, 25, WHITE);
tft.setTextSize(2);
tft.setTextColor(GREEN);
tft.setCursor(X_A2 - 11, Y_A2 + 188);
tft.print("SET");
tft.drawRect(X_A2 + 39, Y_A2 + 182, 50, 25, WHITE);
tft.setTextColor(ORANGE);
tft.setCursor(X_A2 + 47, Y_A2 + 188);
tft.print("Clr");

// Draw date check box
tft.drawRect(X_A2 + 11, Y_A2 + 157, 20, 20, WHITE);

// Draw current settings date
tft.drawRect(X_A2 + 39, Y_A2 + 157, 50, 20, WHITE);

// Draw plus and minus sighn
tft.drawRect(X_A2 + 93, Y_A2 + 182, 60, 20, RED);
tft.setCursor(X_A2 + 98, Y_A2 + 182);
tft.setTextSize(3);
tft.setTextColor(RED);
tft.print(" + ");
tft.drawRect(X_A2 + 93, Y_A2 + 157, 60, 20, BLUE);
tft.setCursor(X_A2 + 98, Y_A2 + 156);
tft.setTextColor(BLUE);
```

```
  tft.print(" - ");


}


// Reset alarm for the next day in the list
void resetAlarmWhenDoW () { // Reset alarm for the next selected day in a week
  if (A1Dy) {
    //  Serial.print("Day of the week: ");
    //  Serial.println(rtc.getDoW());
    //  Serial.print("Alarm set day: ");
    //  Serial.println(A1Day);
    if (rtc.getDoW() != A1Day) {
      for (int i = rtc.getDoW(); i <= 7; i++) {
        if (i == 8) { // Difference in logic. We must do this if we want to include day 7
          i = i - 1;
        }
        //     Serial.print("EEProm alterantion #");
        //     Serial.print(i);
        //     Serial.print(": ");
        //     Serial.print(EEPROM.get(eeAddressAlarmOne + i, alarmOneWeek[i]));
        if (EEPROM.get(eeAddressAlarmOne + i - 1, alarmOneWeek[i - 1]) == true) {
          setAlarm(1, i, A1Hour, A1Minute, A1Second, A1Bits , A1Dy, A1h12 , A1PM);
          // 1 - Which alarm (1 or 2)
          // 2 - Day of the week or Date
          // 3 - Hour
          // 4 - Minute
          // 5 - Seconds
          // 6 - 0x0 Alarm byte
          // 7 - True to set day of the week. False to set alarm for a specific date in a month.
          // 8 - True for 12Hr format and false for 24 Hr
          // 9 - True for PM and false for AM
          // Serial.print("Alarm one was rest to the next day");
          break;
        }
      }
    }
  }

  if (A2Dy) {
    if (rtc.getDoW() != A2Day ) {
      for (int i = rtc.getDoW(); i <= 7; i++) {
        if (i == 8) {
          i = i - 1;
        }
        if (EEPROM.get(eeAddressAlarmTwo + i - 1, alarmTwoWeek[i - 1]) == true) {
          setAlarm(2, i, A2Hour, A2Minute, A1Second, A2Bits , A2Dy, A2h12 , A2PM); // Ignore seconds
          // 1 - Which alarm (1 or 2)
          // 2 - Day of the week or Date
          // 3 - Hour
          // 4 - Minute
          // 5 - Seconds // Ignore for A2. Second alarm does not have seconds.
          // 6 - 0x0 Alarm byte
          // 7 - True to set day of the week. False to set alarm for a specific date in a month.
          // 8 - True for 12Hr format and false for 24 Hr
          // 9 - True for PM and false for AM
          // setAlarm(2, 5, 12, 28, 30, 0x0 , true, false , false);
          break;
        }
      }
    }
  }
}

// Determins a color for each checkmark and draws it on a screen if True
void checkDoW() {
  // Check stored values for an alarm One
  if (alarmOneWeek[0]) {
    drawCheckMarkRed(X_A1 - 39, Y_A1 + 134);
  }
  if (alarmOneWeek[1]) {
    drawCheckMark(X_A1 - 17, Y_A1 + 134);
  }
  if (alarmOneWeek[2]) {
    drawCheckMark(X_A1 + 5, Y_A1 + 134);
  }
  if (alarmOneWeek[3]) {
    drawCheckMark(X_A1 + 27, Y_A1 + 134);
  }
  if (alarmOneWeek[4]) {
    drawCheckMark(X_A1 + 49, Y_A1 + 134);
  }
  if (alarmOneWeek[5]) {
    drawCheckMark(X_A1 + 71, Y_A1 + 134);
  }
  if (alarmOneWeek[6]) {
    drawCheckMarkRed(X_A1 + 93, Y_A1 + 134);
  }
}

// Same here.
void checkDoW2() {
  if (alarmTwoWeek[0]) {
    drawCheckMarkRed(X_A2 - 39, Y_A2 + 134);
  }
  if (alarmTwoWeek[1]) {
    drawCheckMark(X_A2 - 17, Y_A2 + 134);
  }
  if (alarmTwoWeek[2]) {
    drawCheckMark(X_A2 + 5, Y_A2 + 134);
  }
  if (alarmTwoWeek[3]) {
    drawCheckMark(X_A2 + 27, Y_A2 + 134);
  }
  if (alarmTwoWeek[4]) {
    drawCheckMark(X_A2 + 49, Y_A2 + 134);
  }
  if (alarmTwoWeek[5]) {
    drawCheckMark(X_A2 + 71, Y_A2 + 134);
  }
  if (alarmTwoWeek[6]) {
    drawCheckMarkRed(X_A2 + 93, Y_A2 + 134);
  }
}

// Draw play buton
void drawPlay() {
  extern const uint8_t ButtonPlay[4096];
  int  pos_XPLAY = 241;
  int  pos_YPLAY = 177;
  tft.setAddrWindow(pos_XPLAY, pos_YPLAY, pos_XPLAY + 63, pos_YPLAY + 64);
  tft.pushColors(ButtonPlay, 4096, 1);
}

// Activate alarm music
void activateMusicIfAlarm() {
  if ((rtc.checkIfAlarm(1)) || (rtc.checkIfAlarm(2))) {
    mp3.setVolume(30);
    mp3.play();
    playBackStatus = 1;
    if (currentPage == 5) {
      tft.fillRect(209, 129, 63, 62, BLACK);
      drawPause(208, 128);
    }
  }
}
```

```
// Setup. All the good stuff is here
void setup() {


  //  Begin serial
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }


  // MP3 Setup
  mp3Serial.begin(9600);  // BY8001 set to 9600 baud (required)
  mp3.setup(mp3Serial); // tell BY8001 library which serial port to use.
  delay(800);  // allow time for BY8001 cold boot; may adjust depending on flash storage size


  // MP3 Settings
  mp3.setVolume(vol); // Default volume
  playBackStatus = 0; // Default we are not playing any music
  mp3.stopPlayback(); // Just in case MP3 was playing before reboot. // Because of the use of a
secondary power suplly mp3 never resets unless both power cords are out.


  // Begin wire library instamnce for Real Time Clock
  Wire.begin();


  // Mercury tilt switch
  pinMode(tiltSwitch, INPUT);


  // randomSeed(rtc.getSecond() + rtc.getHour(h12, PM) + rtc.getMinute()); // Really random SEED !!!
My invention!
  // randomSeed(rtc.getHour(h12, PM) + rtc.getMinute()); // Moderated (Gives you one minute to swap
battery's if needed.
  //randomSeed(rtc.getHour(h12, PM)); // nerrow
  //randomSeed(1); // Not random seed
  // pinMode(A6, INPUT); // SEED trough open pin
  // randomSeed(analogRead(A6)); // Analog SEED
  // Setup TFT screen
  TFT_Setup();

  // Boot in a Home Screen mode
  currentPage = 0;

  // Initiation of RTC objects;
  // rtc.getHour(h12, PM); // This line is here to get h12 and PM values
  // set_Clock(2, 56, 30, true); // True for 12Hr mode. Upload Hours( First integer) an 24 hour format
even for 12hr mod.
  // Add more 3 to 2 min of upload time. On the first boot.  Last one is h12 state. false for 24 HR
  // set_Date(12, 18, 17, 1); // Last one is the day of the week 1 = Sunday
  // setAlarm(1, 1, 01, 40, 00, 0x0 , true, true , false);
  // setAlarm(2, 1, 07, 01, 01, 0x0 , false, true , false);
  // rtc.setHour(2); // To set Only 24 hour. int please
  // rtc.setMinute(40);
  // rtc.setDoW(7);
  // 1 - Which alarm (1 or 2)
  // 2 - Day of the week or Date
  // 3 - Hour
  // 4 - Minute
  // 5 - Seconds
  // 6 - 0x0 Alarm byte
  // 7 - True to set day of the week. False to set alarm for a specific date in a month.
  // 8 - True for 12Hr format and false for 24 Hr
  // 9 - True for PM and false for AM
  //
  // For loop for debagging EEPROM
  //    for(int i = 0; i <=6; i++){
  //      alarmOneWeek[i] = false; // true
  //      EEPROM.write(eeAddressAlarmOne + i, alarmOneWeek[i]);
  //      Serial.println(EEPROM.get(eeAddressAlarmOne + i, alarmOneWeek[i]));
  //    }
  // delay(800);
  // mp3.playTrackFromFolder(00, 001); // Play music folder 0 file 1



  // Get backround from a SD card
  bool good = SD.begin(SD_CS);
  if (!good) {
    Serial.print(F("cannot start SD"));
    while (1);
  }
  root = SD.open(namebuf);
  pathlen = strlen(namebuf);

  // Get current alarm status
  getAlarm(A1Day, A1Hour, A1Minute, A1Second, A1Bits, A1Dy, A1h12, A1PM, A2Day, A2Hour,
A2Minute, A2Bits, A2Dy, A2h12, A2PM);

  // Draw home screen
  drawHomeScreen();

}

// Im using void to distinguish different states of the currentPage integer to navigate menu back and
forth.
void loop() {

  // Main screen
  if (currentPage == 0) {
    drawAlarmStatus();
    dow(); // Update swich string
    drawDayOfTheWeek(); // Draw new day of the week
    drawTemp();
    drawDate();
    drawHomeClock();
```

```
   // Read touch screen input

   touch_Screen_Read();


   // Coordinates of a Media button

   int  pos_X_MPB = 50;

   int  pos_Y_MPB = 170;


   // If we press media button

   if ((ypos >= pos_Y_MPB) && (ypos <= pos_Y_MPB + 65) && (xpos >= pos_X_MPB) && (xpos <=
pos_X_MPB + 65)) {

     // Zero all data is used in a next screen

     zeroAllData();

     // Set sceren black

     tft.fillScreen(BLACK); // Sets the background color of the area where the text will be printed to black

     // Change scren count

     draw_Media_Screen();

     currentPage = 2;

   }


   // Coordinates of a paint button

   int pos_X_PB = 207;

   int pos_Y_PB = 170;


   // If we press paint button

   if ((ypos >= pos_Y_PB) && (ypos <= pos_Y_PB + 65) && (xpos >= pos_X_PB) && (xpos <= pos_X_PB +
65)) {

     // Change scren count

     currentPage = 1;

     // Zero all data is used in a next screen

     zeroAllData();

     // Draw color selection and a back button

     paint_Setup();

   }
```

```
  // Coordinates of an Alarm button
  int pos_X_AB = 365;
  int pos_Y_AB = 170;

  // If we press Alarm button
  if ((ypos >= pos_Y_AB) && (ypos <= pos_Y_AB + 65) && (xpos >= pos_X_AB) && (xpos <= pos_X_AB +
65)) {
    // Zero all data is used in a next screen
    zeroAllData();
    // Set sceren black
    tft.fillScreen(BLACK); // Sets the background color of the area where the text will be printed to black
    // Get both alarms
    getAlarm(A1Day, A1Hour, A1Minute, A1Second, A1Bits, A1Dy, A1h12, A1PM, A2Day, A2Hour,
A2Minute, A2Bits, A2Dy, A2h12, A2PM);
    getAlarmWeeksFromEEPROM(); // Get days of the week from EEPROM // Because of a random seed
you have to reset days of the week beacouse they drown randomply in EEPROm on a first boot

    // Set new DoW
    newA1Dy = A1Dy;
    newA2Dy = A2Dy;

    // Get last known alarm hour minute and a second
    newA1Hour = A1Hour;
    newA1Minute = A1Minute;
    newA2Hour = A2Hour;
    newA2Minute = A2Minute;

    // Copy old PM h12 to the new alarm
    newA1h12 = A1h12;
    newA1PM = A1PM;
    newA2h12 = A2h12;
    newA2PM = A2PM;
    newA1Date = A1Day;
    newA1Day = A1Day;
    newA2Date = A2Day;
```

```
newA2Day = A2Day;

// Setting alarm control variables
newHourSelector == false;
newMinuteSelector == false;
newHour2Selector == false;
newMinute2Selector == false;

tft.setTextSize(2);
tft.setTextColor(PINK);
tft.setCursor(X_A1 + 50, Y_A1 + 159);

// Draw current settings Alarm One
if (!A1Dy) { // Draw selection of the day in a curent month
  drawCheckMarkWhite(X_A1 + 13, Y_A1 + 159);
  newDoWSelector = true;

  if ((newA1Date <= 31) && (newA1Date >= 10)) {
    tft.print(newA1Date);
  }
  else if ((newA1Date < 10) && (newA1Date >= 1)) {
    tft.print('0');
    tft.setCursor(X_A1 + 62, Y_A1 + 159);
    tft.print(newA1Date);
  }
}
else
{
  checkDoW();
  newDoWSelector = false;
  switch (newA1Day) {
    case 1:
      day_Of_The_Week = "Sun";
      break;
  case 2:
    day_Of_The_Week = "Mon";
    break;
  case 3:
    day_Of_The_Week = "Tue";
    break;
  case 4:
    day_Of_The_Week = "Wed";
    break;
  case 5:
    day_Of_The_Week = "Thu";
    break;
  case 6:
    day_Of_The_Week = "Fri";
    break;
  case 7:
    day_Of_The_Week = "Sat";
    break;
 }
 tft.print(day_Of_The_Week);
}

// Set Cursor
tft.setCursor(X_A2 + 50, Y_A2 + 159);

// Draw current settings Alarm Two
if (!A2Dy) { // Draw selection of the day in a curent month
 drawCheckMarkWhite(X_A2 + 13, Y_A2 + 159);
 newDoW2Selector = true;

 if ((newA2Date <= 31) && (newA2Date >= 10)) {
  tft.print(newA2Date);
 }
 else if ((newA2Date < 10) && (newA2Date >= 1)) {
      tft.print('0');
      tft.setCursor(X_A2 + 62, Y_A2 + 159);
      tft.print(newA2Date);
    }
  }
  else
  {
    checkDoW2();
    newDoW2Selector = false;
    switch (newA2Day) {
      case 1:
        day_Of_The_Week = "Sun";
        break;
      case 2:
        day_Of_The_Week = "Mon";
        break;
      case 3:
        day_Of_The_Week = "Tue";
        break;
      case 4:
        day_Of_The_Week = "Wed";
        break;
      case 5:
        day_Of_The_Week = "Thu";
        break;
      case 6:
        day_Of_The_Week = "Fri";
        break;
      case 7:
        day_Of_The_Week = "Sat";
        break;
    }
    tft.print(day_Of_The_Week);
  }
```

```
    // Draw alarm screen
    draw_Alarm_Screen();

    // Change scren count
    currentPage = 3;
  }
}

// Paint screen
if (currentPage == 1) {
  zeroAllData();
  touch_Screen_Read();
  if (xpos != -1) {
    paint_Loop();
  }
}

// Media screen
if (currentPage == 2) {
  drawTemp();
  drawDate();
  drawDayOfTheWeek();
  drawSmallClock();
  touch_Screen_Read();

  // If we press radio button
  if ((xpos >= 190) && (xpos <= 260) && (ypos >= 145) && (ypos <= 215)) {
    xpos = -1;
    ypos = -1;
    // Zero all data is used in a next screen
    zeroAllData();
    // Set sceren black
  tft.fillScreen(BLACK); // Sets the background color of the area where the text will be printed to black
  // Draw Radio screen
  draw_Radio_Screen();
  currentPage = 4;
}

// If we press back button
if ((ypos > tft.height() - 40) && (xpos < 40)) {
  zeroAllData();
  drawHomeScreen();
  currentPage = 0;
}

// If we press MP3 button
if ((xpos >= 65) && (xpos <= 135) && (ypos >= 140) && (ypos <= 215)) {
  xpos = -1;
  ypos = -1;
  // Zero all data is used in a next screen
  zeroAllData();
  // Set sceren black
  tft.fillScreen(BLACK); // Sets the background color of the area where the text will be printed to black
  drawBackButton();
  // Draw Mp3 screen
  mp3_Player_Screen();
  currentPage = 5;
}

// If we press Bluetooth button
if ((xpos >= 320) && (xpos <= 370) && (ypos >= 140) && (ypos <= 215)) {
  xpos = -1;
  ypos = -1;
  // Zero all data is used in a next screen
  zeroAllData();
  // Set sceren black
```

```
    tft.fillScreen(BLACK); // Sets the background color of the area where the text will be printed to black

    drawBackButton();

    // Draw Blurtooth screen

    //mp3_Player_Screen();

    currentPage = 6;

   }


 }


 // Alarm screen

 if (currentPage == 3) {

  // Update screen data

  drawTemp();

  drawDate();

  dow();

  drawDayOfTheWeek();

  drawSmallClock();

  touch_Screen_Read();


  // If we press Minus button while alarm is set for days of the month

  if ((!newMinuteSelector) && (!newHourSelector) && (newDoWSelector) && (xpos >= X_A1 + 75) &&
(xpos <= X_A1 + 140) && (ypos >= Y_A1 + 157) && (ypos <= Y_A1 + 180)) {

    xpos = -1;

    ypos = -1;

    newAlarmOne = true;

    newA1Date--;

    tft.fillRect(X_A1 + 50, Y_A1 + 158, 22, 18, BLACK);

    tft.setTextColor(PINK);

    tft.setCursor(X_A1 + 50, Y_A1 + 159);



    if ((newA1Date <= 31) && (newA1Date >= 10)) {

      tft.print(newA1Date);

    }
    else if ((newA1Date < 10) && (newA1Date >= 1)) {
      tft.print('0');
      tft.setCursor(X_A1 + 62, Y_A1 + 159);
      tft.print(newA1Date);
    }
    else if (newA1Date == 0) {
      newA1Date = 31;
      tft.print(newA1Date);
    }

    delay(t);
  }

  // If we press Plus button while alarm is set for days of the month
  if ((!newMinuteSelector) && (!newHourSelector) && (newDoWSelector) && (xpos >= X_A1 + 75) &&
(xpos <= X_A1 + 140) && (ypos >= Y_A1 + 180) && (ypos <= Y_A1 + 210)) {
    xpos = -1;
    ypos = -1;
    newAlarmOne = true;
    newA1Date++; // Counter up
    tft.fillRect(X_A1 + 50, Y_A1 + 158, 22, 18, BLACK);
    tft.setTextColor(PINK);
    tft.setCursor(X_A1 + 50, Y_A1 + 159);
    tft.fillRect(X_A1 + 50, Y_A1 + 158, 22, 18, BLACK);

    if ((newA1Date <= 31) && (newA1Date >= 10)) {
      tft.print(newA1Date);
    }
    else if ((newA1Date < 10) && (newA1Date >= 1)) {
      tft.print('0');
      tft.setCursor(X_A1 + 62, Y_A1 + 159);
      tft.print(newA1Date);
    }
    else if (newA1Date == 32) {
      tft.print('0');
      tft.setCursor(X_A1 + 62, Y_A1 + 159);
      newA1Date = 1;
      tft.print(newA1Date);
    }

    delay(t);
  }

  // If we click at the Hours
  if ((xpos >= X_A1 - 20) && (xpos <= X_A1 + 30) && (ypos >= Y_A1 + 94) && (ypos <= Y_A1 + 130)) {
    xpos = -1;
    ypos = -1;
    newHourSelector = true;
    newMinuteSelector = false;
    newAlarmOne = true;
    newDoWSelector = false;
  }

  // If we click at the Minutess
  if ((xpos >= X_A1 + 25) && (xpos <= X_A1 + 85) && (ypos >= Y_A1 + 94) && (ypos <= Y_A1 + 130)) {
    xpos = -1;
    ypos = -1;
    newHourSelector = false;
    newMinuteSelector = true;
    newAlarmOne = true;
    newDoWSelector = false;
  }

  // If we press Minus button while alarm selector is at Hours
  if ((!newDoWSelector) && (!newMinuteSelector) && (newHourSelector) && (xpos >= X_A1 + 75) &&
(xpos <= X_A1 + 140) && (ypos >= Y_A1 + 157) && (ypos <= Y_A1 + 180)) {
    xpos = -1;
    ypos = -1;
```

```
delay(t);
tft.setCursor(X_A1 - 11, Y_A1 + 94);
tft.setTextColor(BLUE);
tft.setTextSize(4);
if (newA1h12) {
  tft.fillRect(X_A1 - 11, Y_A1 + 94, 44, 28, BLACK);
  newA1Hour--;
  if ((newA1Hour >= 1) && (newA1Hour < 10)) {
    tft.print('0');
    tft.setCursor(X_A1 + 11, Y_A1 + 94);
    tft.print(newA1Hour);
  }
  else if ((newA1Hour >= 10) && (newA1Hour <= 12)) {
    tft.print(newA1Hour);
  }
  else if (newA1Hour == 0) {
    newA1Hour = 12;
    newA1PM = !newA1PM;
    tft.print(newA1Hour);
    tft.fillRect(X_A1 - 49, Y_A1 + 93, 22, 15, BLACK);
    tft.setCursor(X_A1 - 49, Y_A1 + 93);
    tft.setTextSize(2);
    tft.setTextColor(RED);
    if (newA1PM) {
      tft.print("PM");
    }
    else tft.print("AM");
  }
}
else { // 24H
  tft.fillRect(X_A1, Y_A1 + 94, 44, 28, BLACK);
  if (newA1Hour > 0) {
    newA1Hour--;
    tft.setCursor(X_A1, Y_A1 + 94);
```

```
        if (newA1Hour < 10) {
          tft.print('0');
          tft.setCursor(X_A1 + 24, Y_A1 + 94);
          tft.print(newA1Hour);
        }
        else tft.print(newA1Hour); // Larger than 10
      }
      else if (newA1Hour == 0) {
        tft.setCursor(X_A1, Y_A1 + 94);
        newA1Hour = 23;
        tft.print(newA1Hour);
      }
    }
  }

  // If we press Plus button while alarm is selector is at Hours
  if ((!newDoWSelector) && (!newMinuteSelector) && (newHourSelector) && (xpos >= X_A1 + 75) &&
(xpos <= X_A1 + 140) && (ypos >= Y_A1 + 180) && (ypos <= Y_A1 + 210)) {
    xpos = -1;
    ypos = -1;
    delay(t);
    tft.setCursor(X_A1 - 11, Y_A1 + 94);
    tft.setTextColor(BLUE);
    tft.setTextSize(4);
    if (newA1h12) {
      tft.fillRect(X_A1 - 11, Y_A1 + 94, 44, 28, BLACK);
      newA1Hour++;
      if ((newA1Hour >= 1) && (newA1Hour < 10)) {
        tft.print('0');
        tft.setCursor(X_A1 + 11, Y_A1 + 94);
        tft.print(newA1Hour);
      }
      else if ((newA1Hour >= 10) && (newA1Hour <= 12)) {
        tft.print(newA1Hour);
  }
  else if (newA1Hour == 13) {
    newA1Hour = 1;
    newA1PM = !newA1PM;
    tft.print('0');
    tft.setCursor(X_A1 + 11, Y_A1 + 94);
    tft.print(newA1Hour);
    tft.fillRect(X_A1 - 49, Y_A1 + 93, 22, 15, BLACK);
    tft.setCursor(X_A1 - 49, Y_A1 + 93);
    tft.setTextSize(2);
    tft.setTextColor(RED);
    if (newA1PM) {
      tft.print("PM");
    }
    else tft.print("AM");
  }
}
else { // 24 hr format
  tft.fillRect(X_A1, Y_A1 + 94, 44, 28, BLACK);
  tft.setCursor(X_A1, Y_A1 + 94);
  if (newA1Hour < 23) {
    newA1Hour++;
    if (newA1Hour < 10) {
      tft.print('0');
      tft.setCursor(X_A1 + 24, Y_A1 + 94);
      tft.print(newA1Hour);
    }
    else tft.print(newA1Hour); // Larger than 10
  }
  else if (newA1Hour == 23) {
    newA1Hour = 0;
    tft.print('0');
    tft.setCursor(X_A1 + 24, Y_A1 + 94);
    tft.print(newA1Hour);
```

```
      }
    }
  }

  // If we press Minus button while alarm selector is at Minutes
  if ((!newDoWSelector) && (newMinuteSelector) && (!newHourSelector) && (xpos >= X_A1 + 75) &&
(xpos <= X_A1 + 140) && (ypos >= Y_A1 + 157) && (ypos <= Y_A1 + 180)) {
    xpos = -1;
    ypos = -1;
    delay(t);
    tft.setTextColor(BLUE);
    tft.setTextSize(4);
    if (!newA1h12) {
      tft.fillRect(X_A1 + 55, Y_A1 + 94, 44, 28, BLACK);
      tft.setCursor(X_A1 + 55, Y_A1 + 94);
    }
    else {
      tft.fillRect(X_A1 + 43, Y_A1 + 94, 44, 28, BLACK);
      tft.setCursor(X_A1 + 43, Y_A1 + 94);
    }
    newA1Minute--;
    if (newA1Minute == -1) {
      newA1Minute = 59;
      tft.print(newA1Minute);
    }
    else if ((newA1Minute >= 0) && (newA1Minute <= 9)) {
      tft.print('0');
      if (!newA1h12) {
        tft.setCursor(X_A1 + 78, Y_A1 + 94);
      }
      else {
        tft.setCursor(X_A1 + 66, Y_A1 + 94);
      }
      tft.print(newA1Minute);
    }
    else if ((newA1Minute >= 10) && (newA1Minute <= 60)) {
      tft.print(newA1Minute);
    }


  }


  // If we press Plus button while alarm selector is at Minutes
  if ((!newDoWSelector) && (newMinuteSelector) && (!newHourSelector) && (xpos >= X_A1 + 75) &&
(xpos <= X_A1 + 140) && (ypos >= Y_A1 + 180) && (ypos <= Y_A1 + 210)) {
    xpos = -1;
    ypos = -1;
    delay(t);
    tft.setTextColor(BLUE);
    tft.setTextSize(4);
    if (!newA1h12) {
      tft.fillRect(X_A1 + 55, Y_A1 + 94, 44, 28, BLACK);
      tft.setCursor(X_A1 + 55, Y_A1 + 94);
    } else {
      tft.fillRect(X_A1 + 43, Y_A1 + 94, 44, 28, BLACK);
      tft.setCursor(X_A1 + 43, Y_A1 + 94);
    }


    newA1Minute++;
    if (newA1Minute == 60) {
      newA1Minute = 0;
      tft.print('0');
      if (!newA1h12) {
        tft.setCursor(X_A1 + 78, Y_A1 + 94);
      } else {
        tft.setCursor(X_A1 + 66, Y_A1 + 94);
      }
      tft.print(newA1Minute);
    }
```

```
 else if ((newA1Minute >= 10) && (newA1Minute < 60 )) {
  tft.print(newA1Minute);
 }
 else if ((newA1Minute >= 0) && (newA1Minute < 10)) {
  tft.print('0');
  if (!newA1h12) {
   tft.setCursor(X_A1 + 78, Y_A1 + 94);
  } else {
   tft.setCursor(X_A1 + 66, Y_A1 + 94);
  }
  tft.print(newA1Minute);
 }
}

// if we press Date to DoW switch
if ((xpos >= X_A1 - 3) && (xpos <= X_A1 + 15) && (ypos >= Y_A1 + 157) && (ypos <= Y_A1 + 180)) {
 xpos = -1;
 ypos = -1;
 newA1Dy = !newA1Dy;
 newAlarmOne = true;
 newHourSelector = false;
 newMinuteSelector = false;
 tft.fillRect(X_A1 + 45, Y_A1 + 158, 40, 18, BLACK); // Draw a balck square over the last known value

 if (!newA1Dy) { // Draw selection of the day in a curent month

  // Turn on DoW selector
  newDoWSelector = true;
  drawCheckMarkWhite(X_A1 + 13, Y_A1 + 159);
  tft.setTextColor(PINK);
  tft.setCursor(X_A1 + 50, Y_A1 + 159);
  if ((newA1Date <= 31) && (newA1Date >= 10)) {
   tft.print(newA1Date);
  }
```

```
    else if ((newA1Date < 10) && (newA1Date >= 1)) {
      tft.print('0');
      tft.setCursor(X_A1 + 62, Y_A1 + 159);
      tft.print(newA1Date);
    }
    tft.fillRect(X_A1 - 40, Y_A1 + 133, 18, 18, BLACK);
    tft.fillRect(X_A1 - 18, Y_A1 + 133, 18, 18, BLACK);
    tft.fillRect(X_A1 + 4, Y_A1 + 133, 18, 18, BLACK);
    tft.fillRect(X_A1 + 26, Y_A1 + 133, 18, 18, BLACK);
    tft.fillRect(X_A1 + 48, Y_A1 + 133, 18, 18, BLACK);
    tft.fillRect(X_A1 + 70, Y_A1 + 133, 18, 18, BLACK);
    tft.fillRect(X_A1 + 92, Y_A1 + 133, 18, 18, BLACK);

  }
  else {  // Draw selection of the week

    // Turn off DoW selector
    newDoWSelector = false;

    // Draw check marks
    checkDoW(); // Draw checkmarks

    // Draw black box over both areas and reset counters
    tft.fillRect(X_A1 + 12, Y_A1 + 158, 18, 18, BLACK);
  }
  delay(t);
}

// If we press set button
if ((xpos >= X_A1 - 33) && (xpos <= X_A1 + 14) && (ypos >= Y_A1 + 182) && (ypos <= Y_A1 + 207)) {
  // Zero touchscreen
  xpos = -1;
  ypos = -1;
  for (int i = rtc.getDoW(); i <= 7; i++) {
```

```
  if (i == 8) {
    i = i - 1;
  }
  if (alarmOneWeek[i - 1] == true) {
    newA1Day = i;
    break;
  }
  else if (i == 7) {
    for (int i = 1; i <= 7; i++) {
      if (i == 8) {
        i = i - 1;
      }
      if (alarmOneWeek[i - 1] == true) {
        newA1Day = i;
        break;
      }
    }
  }
}

if (newAlarmOne) { // If we changet anythings it will be true
  newDoW2Selector = false; // Flag off
  switch (newA1Day) {
    case 1:
      day_Of_The_Week = "Sunday";
      break;
    case 2:
      day_Of_The_Week = "Monday";
      break;
    case 3:
      day_Of_The_Week = "Tuesday";
      break;
    case 4:
      day_Of_The_Week = "Wednesday";
      break;
    case 5:
      day_Of_The_Week = "Thursday";
      break;
    case 6:
      day_Of_The_Week = "Friday";
      break;
    case 7:
      day_Of_The_Week = "Saturday";
      break;
  }
  tft.fillRect(X_A1 + 42, Y_A1 + 159, 45, 16, BLACK);
  tft.setCursor(X_A1 + 45, Y_A1 + 159);

  // Set Cursor, color and size
  tft.setTextSize(2);
  tft.setTextColor(PINK);

  if (newA1Dy) {
    String newA1SDay = day_Of_The_Week.substring(0, 3);
    tft.print(newA1SDay);
    // set Alarm
     if((newA1h12) && (newA1PM) && ( newA1Hour <= 12)) newA1Hour = newA1Hour + 12;
    setAlarm(1, newA1Day, newA1Hour, newA1Minute, A1Second, A1Bits, newA1Dy, newA1h12 ,
newA1PM);
  }
  else {
    // Restore position
    tft.setCursor(X_A1 + 50, Y_A1 + 159);
    drawCheckMarkWhite(X_A1 + 13, Y_A1 + 159);
    newDoWSelector = true;

    if ((newA1Date <= 31) && (newA1Date >= 10)) {
      tft.print(newA1Date);
```

```
        }
        else if ((newA1Date < 10) && (newA1Date >= 1)) {
         tft.print('0');
         tft.setCursor(X_A1 + 62, Y_A1 + 159);
         tft.print(newA1Date);
        }

        // Set alarm
        if((newA1h12) && (newA1PM) && ( newA1Hour <= 12)) newA1Hour = newA1Hour + 12;
        setAlarm(1, newA1Date, newA1Hour, newA1Minute, A1Second, A1Bits, newA1Dy, newA1h12 ,
newA1PM);
       }

       // Change Status
       rtc.turnOnAlarm(1); // Turn alarm one on
       tft.setCursor(X_A1 + 65, Y_A1 + 51);
       tft.fillRect(X_A1 + 65, Y_A1 + 51, 34, 14, BLACK);
       checkAlarmStatus(1);

      }
    delay(t*10);
    }

    // If we press clear button
    if ((xpos >= X_A1 + 24) && (xpos <= X_A1 + 74) && (ypos >= Y_A1 + 182) && (ypos <= Y_A1 + 207)) {
     // Zero touchscreen
     xpos = -1;
     ypos = -1;
     rtc.turnOffAlarm(1);
     tft.setCursor(X_A1 + 65, Y_A1 + 51);
     tft.fillRect(X_A1 + 65, Y_A1 + 51, 34, 14, BLACK);
     checkAlarmStatus(1);

     // Draw one instead of current alarm 1 hour
```

```
newA1Hour = 1;
newAlarmOne = false;

if (!A1h12) { // If 24 hours format

  tft.setCursor(X_A1, Y_A1 + 94);
} else tft.setCursor(X_A1 - 11, Y_A1 + 94);

tft.setTextColor(BLUE);
tft.setTextSize(4);
if (!A1h12) { // If 24 hours format
  tft.fillRect(X_A1, Y_A1 + 94, 44, 28, BLACK);
} else tft.fillRect(X_A1 - 11, Y_A1 + 94, 44, 28, BLACK);

tft.print('0');

if (!A1h12) { // If 24 hours format

  tft.setCursor(X_A1 + 24, Y_A1 + 94);
} else tft.setCursor(X_A1 + 13, Y_A1 + 94);

tft.print(newA1Hour);

// Draw one instead of current alarm 1 minute
newA1Minute = 1;

if (!A1h12) {
  tft.fillRect(X_A1 + 55, Y_A1 + 94, 44, 28, BLACK);
  tft.setCursor(X_A1 + 55, Y_A1 + 94);
} else {
  tft.fillRect(X_A1 + 43, Y_A1 + 94, 44, 28, BLACK);
  tft.setCursor(X_A1 + 43, Y_A1 + 94);
}
tft.print('0');
```

```
  if (!A1h12) {
    tft.setCursor(X_A1 + 79, Y_A1 + 94);
  } else tft.setCursor(X_A1 + 67, Y_A1 + 94);
  tft.print(newA1Minute);

  // Turn off existing check marks
  tft.fillRect(X_A1 - 40, Y_A1 + 133, 18, 18, BLACK);
  tft.fillRect(X_A1 - 18, Y_A1 + 133, 18, 18, BLACK);
  tft.fillRect(X_A1 + 4, Y_A1 + 133, 18, 18, BLACK);
  tft.fillRect(X_A1 + 26, Y_A1 + 133, 18, 18, BLACK);
  tft.fillRect(X_A1 + 48, Y_A1 + 133, 18, 18, BLACK);
  tft.fillRect(X_A1 + 70, Y_A1 + 133, 18, 18, BLACK);
  tft.fillRect(X_A1 + 92, Y_A1 + 133, 18, 18, BLACK);

  // Draw black box over both areas and reset counters
  tft.fillRect(X_A1 + 12, Y_A1 + 158, 18, 18, BLACK);
  tft.fillRect(X_A1 + 45, Y_A1 + 158, 42, 18, BLACK);
  newA1Date = 1;
  newA1Day = 1;

  newA1Dy = true;
  for (int i = 0; i <= 6; i++) {
    alarmOneWeek[i] = false;
    EEPROM.write(eeAddressAlarmOne + i, alarmOneWeek[i]);
  }

}

// If we pressing back button
if ((ypos > tft.height() - 35) && (xpos <= 20)) {
  zeroAllData();
  currentPage = 0;
  drawHomeScreen();
  }

  if (newA1Dy) { // Are we setting up alarm for the days of the week? operate checkmarks. If false than
we setting it up for the date in a month

    // If we click in the first check Box
    if ((xpos >= X_A1 - 50) && (xpos <= X_A1 - 34) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 154)) {
      newAlarmOne = true;
      if (alarmOneWeek[0] == false) {
        // Zero touchscreen
        xpos = -1;
        ypos = -1;
        drawCheckMarkRed(X_A1 - 39, Y_A1 + 134);
        alarmOneWeek[0] = true;
        delay(t);
      } else {
        // Zero touchscreen
        xpos = -1;
        ypos = -1;
        tft.fillRect(X_A1 - 40, Y_A1 + 133, 18, 18, BLACK);
        alarmOneWeek[0] = false;
        delay(t);
      }
    }

    // If we click in the second check Box
    if ((xpos >= X_A1 - 33) && (xpos <= X_A1 - 17) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 152)) {
      newAlarmOne = true;
      // If button is on or off
      if (alarmOneWeek[1] == false) {
        // Zero touchscreen
        xpos = -1;
        ypos = -1;
        drawCheckMark(X_A1 - 17, Y_A1 + 134);
    alarmOneWeek[1] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A1 - 18, Y_A1 + 133, 18, 18, BLACK);
    alarmOneWeek[1] = false;
    delay(t);
  }
}

// If we click in the check Box #3
if ((xpos >= X_A1 - 11) && (xpos <= X_A1 + 7) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 152)) {
  newAlarmOne = true;
  // If button is on or off
  if (alarmOneWeek[2] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A1 + 5, Y_A1 + 134);
    alarmOneWeek[2] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A1 + 4, Y_A1 + 133, 18, 18, BLACK);
    alarmOneWeek[2] = false;
    delay(t);
  }
}

// If we click in the check Box #4
```

```
if ((xpos >= X_A1 + 11) && (xpos <= X_A1 + 29) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 152)) {
  newAlarmOne = true;
  // If button is on or off
  if (alarmOneWeek[3] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A1 + 27, Y_A1 + 134);
    alarmOneWeek[3] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A1 + 26, Y_A1 + 133, 18, 18, BLACK);
    alarmOneWeek[3] = false;
    delay(t);
  }
}

// If we click in the check Box #5
if ((xpos >= X_A1 + 33) && (xpos <= X_A1 + 51) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 152)) {
  newAlarmOne = true;
  // If button is on or off
  if (alarmOneWeek[4] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A1 + 49, Y_A1 + 134);
    alarmOneWeek[4] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A1 + 48, Y_A1 + 133, 18, 18, BLACK);
    alarmOneWeek[4] = false;
    delay(t);
  }
}

// If we click in the check Box #6
if ((xpos >= X_A1 + 55) && (xpos <= X_A1 + 73) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 152)) {
  newAlarmOne = true;
  // If button is on or off
  if (alarmOneWeek[5] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A1 + 71, Y_A1 + 134);
    alarmOneWeek[5] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A1 + 70, Y_A1 + 133, 18, 18, BLACK);
    alarmOneWeek[5] = false;
    delay(t);
  }
}

// If we click in the check Box #7
if ((xpos >= X_A1 + 77) && (xpos <= X_A1 + 95) && (ypos >= Y_A1 + 132) && (ypos <= Y_A1 + 152)) {
  newAlarmOne = true;
  // If button is on or off
  if (alarmOneWeek[6] == false) {
    // Zero touchscreen
```

```
        xpos = -1;
        ypos = -1;
        drawCheckMarkRed(X_A1 + 93, Y_A1 + 134);
        alarmOneWeek[6] = true;
        delay(t);
      } else {
        // Zero touchscreen
        xpos = -1;
        ypos = -1;
        tft.fillRect(X_A1 + 92, Y_A1 + 133, 18, 18, BLACK);
        alarmOneWeek[6] = false;
        delay(t);
      }
    }
  }



//////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////
  // Alarm two set

  // If we click at the Hours
  if ((xpos >= X_A2 - 20) && (xpos <= X_A2 + 30) && (ypos >= Y_A2 + 94) && (ypos <= Y_A2 + 130)) {
    xpos = -1;
    ypos = -1;
    newHour2Selector = true;
    newMinute2Selector = false;
    newAlarmTwo = true;
    newDoW2Selector = false;
  }

  // If we click at the Minutess
  if ((xpos >= X_A2 + 25) && (xpos <= X_A2 + 85) && (ypos >= Y_A2 + 94) && (ypos <= Y_A2 + 130)) {
    xpos = -1;
    ypos = -1;
    newHour2Selector = false;
    newMinute2Selector = true;
    newAlarmTwo = true;
    newDoW2Selector = false;
  }

  // If we press Minus button while alarm selector is at Hours
  if ((!newDoW2Selector) && (!newMinute2Selector) && (newHour2Selector) && (xpos >= X_A2 + 75)
&& (xpos <= X_A2 + 140) && (ypos >= Y_A2 + 157) && (ypos <= Y_A2 + 180)) {
    xpos = -1;
    ypos = -1;
    delay(t);
    tft.setCursor(X_A2 - 11, Y_A2 + 94);
    tft.setTextColor(BLUE);
    tft.setTextSize(4);
    if (newA2h12) {
      tft.fillRect(X_A2 - 11, Y_A2 + 94, 44, 28, BLACK);
      newA2Hour--;
      if ((newA2Hour >= 1) && (newA2Hour < 10)) {
        tft.print('0');
        tft.setCursor(X_A2 + 11, Y_A2 + 94);
        tft.print(newA2Hour);
      }
      else if ((newA2Hour >= 10) && (newA2Hour <= 12)) {
        tft.print(newA2Hour);
      }
      else if (newA2Hour == 0) {
        newA2Hour = 12;
        newA2PM = !newA2PM;
        tft.print(newA2Hour);
        tft.fillRect(X_A2 - 49, Y_A2 + 93, 22, 15, BLACK);
        tft.setCursor(X_A2 - 49, Y_A2 + 93);
        tft.setTextSize(2);
        tft.setTextColor(RED);
        if (newA2PM) {
          tft.print("PM");
        }
        else tft.print("AM");
      }
    }
    else { // 24H
      tft.fillRect(X_A2, Y_A2 + 94, 44, 28, BLACK);
      if (newA2Hour > 0) {
        newA2Hour--;
        tft.setCursor(X_A2, Y_A2 + 94);
        if (newA2Hour < 10) {
          tft.print('0');
          tft.setCursor(X_A2 + 24, Y_A2 + 94);
          tft.print(newA2Hour);
        }
        else tft.print(newA2Hour); // Larger than 10
      }
      else if (newA2Hour == 0) {
        tft.setCursor(X_A2, Y_A2 + 94);
        newA2Hour = 23;
        tft.print(newA2Hour);
      }
    }
  }

  // If we press Plus button while alarm is selector is at Hours
  if ((!newDoW2Selector) && (!newMinute2Selector) && (newHour2Selector) && (xpos >= X_A2 + 75)
&& (xpos <= X_A2 + 140) && (ypos >= Y_A2 + 180) && (ypos <= Y_A2 + 210)) {
    xpos = -1;
    ypos = -1;
    delay(t);
```

```
tft.setCursor(X_A2 - 11, Y_A2 + 94);
tft.setTextColor(BLUE);
tft.setTextSize(4);
if (newA2h12) {
  tft.fillRect(X_A2 - 11, Y_A2 + 94, 44, 28, BLACK);
  newA2Hour++;
  if ((newA2Hour >= 1) && (newA2Hour < 10)) {
    tft.print('0');
    tft.setCursor(X_A2 + 11, Y_A2 + 94);
    tft.print(newA2Hour);
  }
  else if ((newA2Hour >= 10) && (newA2Hour <= 12)) {
    tft.print(newA2Hour);
  }
  else if (newA2Hour == 13) {
    newA2Hour = 1;
    newA2PM = !newA2PM;
    tft.print('0');
    tft.setCursor(X_A2 + 11, Y_A2 + 94);
    tft.print(newA2Hour);
    tft.fillRect(X_A2 - 49, Y_A2 + 93, 22, 15, BLACK);
    tft.setCursor(X_A2 - 49, Y_A2 + 93);
    tft.setTextSize(2);
    tft.setTextColor(RED);
    if (newA2PM) {
      tft.print("PM");
    }
    else tft.print("AM");
  }
}
else { // 24 hr format
  tft.fillRect(X_A2, Y_A2 + 94, 44, 28, BLACK);
  tft.setCursor(X_A2, Y_A2 + 94);
  if (newA2Hour < 23) {
        newA2Hour++;
        if (newA2Hour < 10) {
          tft.print('0');
          tft.setCursor(X_A2 + 24, Y_A2 + 94);
          tft.print(newA2Hour);
        }
        else tft.print(newA2Hour); // Larger than 10
      }
      else if (newA2Hour == 23) {
        newA2Hour = 0;
        tft.print('0');
        tft.setCursor(X_A2 + 24, Y_A2 + 94);
        tft.print(newA2Hour);
      }
    }
  }

  // If we press Minus button while alarm selector is at Minutes
  if ((!newDoW2Selector) && (newMinute2Selector) && (!newHour2Selector) && (xpos >= X_A2 + 75)
&& (xpos <= X_A2 + 140) && (ypos >= Y_A2 + 157) && (ypos <= Y_A2 + 180)) {
    xpos = -1;
    ypos = -1;
    delay(t);
    tft.setTextColor(BLUE);
    tft.setTextSize(4);
    if (!newA2h12) {
      tft.fillRect(X_A2 + 55, Y_A2 + 94, 44, 28, BLACK);
      tft.setCursor(X_A2 + 55, Y_A2 + 94);
    }
    else {
      tft.fillRect(X_A2 + 43, Y_A2 + 94, 44, 28, BLACK);
      tft.setCursor(X_A2 + 43, Y_A2 + 94);
    }
    newA2Minute--;
```

```
      if (newA2Minute == -1) {
        newA2Minute = 59;
        tft.print(newA2Minute);
      }
      else if ((newA2Minute >= 0) && (newA2Minute <= 9)) {
        tft.print('0');
        if (!newA2h12) {
          tft.setCursor(X_A2 + 78, Y_A2 + 94);
        }
        else {
          tft.setCursor(X_A2 + 66, Y_A2 + 94);
        }
        tft.print(newA2Minute);
      }
      else if ((newA2Minute >= 10) && (newA2Minute <= 60)) {
        tft.print(newA2Minute);
      }

    }

    // If we press Plus button while alarm selector is at Minutes
    if ((!newDoW2Selector) && (newMinute2Selector) && (!newHour2Selector) && (xpos >= X_A2 + 75)
  && (xpos <= X_A2 + 140) && (ypos >= Y_A2 + 180) && (ypos <= Y_A2 + 210)) {
      xpos = -1;
      ypos = -1;
      delay(t);
      tft.setTextColor(BLUE);
      tft.setTextSize(4);
      if (!newA2h12) {
        tft.fillRect(X_A2 + 55, Y_A2 + 94, 44, 28, BLACK);
        tft.setCursor(X_A2 + 55, Y_A2 + 94);
      } else {
        tft.fillRect(X_A2 + 43, Y_A2 + 94, 44, 28, BLACK);
        tft.setCursor(X_A2 + 43, Y_A2 + 94);
      }


      newA2Minute++;
      if (newA2Minute == 60) {
        newA2Minute = 0;
        tft.print('0');
        if (!newA2h12) {
          tft.setCursor(X_A2 + 78, Y_A2 + 94);
        } else {
          tft.setCursor(X_A2 + 66, Y_A2 + 94);
        }
        tft.print(newA2Minute);
      }
      else if ((newA2Minute >= 10) && (newA2Minute < 60 )) {
        tft.print(newA2Minute);
      }
      else if ((newA2Minute >= 0) && (newA2Minute < 10)) {
        tft.print('0');
        if (!newA2h12) {
          tft.setCursor(X_A2 + 78, Y_A2 + 94);
        } else {
          tft.setCursor(X_A2 + 66, Y_A2 + 94);
        }
        tft.print(newA2Minute);
      }
    }


    // If we press Minus button while alarm 2 is set for days of the month
    if ((!newMinute2Selector) && (!newHour2Selector) && (newDoW2Selector) && (xpos >= X_A2 + 75)
  && (xpos <= X_A2 + 140) && (ypos >= Y_A2 + 157) && (ypos <= Y_A2 + 180)) {
      xpos = -1;
      ypos = -1;
      newAlarmTwo = true;
      newA2Date--;
    tft.fillRect(X_A2 + 50, Y_A2 + 158, 22, 18, BLACK);

    tft.setTextColor(PINK);

    tft.setCursor(X_A2 + 50, Y_A2 + 159);



    if ((newA2Date <= 31) && (newA2Date >= 10)) {

      tft.print(newA2Date);

    }

    else if ((newA2Date < 10) && (newA2Date >= 1)) {

      tft.print('0');

      tft.setCursor(X_A2 + 62, Y_A2 + 159);

      tft.print(newA2Date);

    }

    else if (newA2Date == 0) {

      newA2Date = 31;

      tft.print(newA2Date);

    }


    delay(t);

  }


  // If we press Plus button while alarm 2 is set for days of the month

  if ((!newMinute2Selector) && (!newHour2Selector) && (newDoW2Selector) && (xpos >= X_A2 + 75)
&& (xpos <= X_A2 + 140) && (ypos >= Y_A2 + 180) && (ypos <= Y_A2 + 210)) {

    xpos = -1;

    ypos = -1;

    newAlarmTwo = true;

    newA2Date++; // Counter up

    tft.fillRect(X_A2 + 50, Y_A2 + 158, 22, 18, BLACK);

    tft.setTextColor(PINK);

    tft.setCursor(X_A2 + 50, Y_A2 + 159);

    tft.fillRect(X_A2 + 50, Y_A2 + 158, 22, 18, BLACK);


    if ((newA2Date <= 31) && (newA2Date >= 10)) {
```

```
    tft.print(newA2Date);
  }
  else if ((newA2Date < 10) && (newA2Date >= 1)) {
    tft.print('0');
    tft.setCursor(X_A2 + 62, Y_A2 + 159);
    tft.print(newA2Date);
  }
  else if (newA2Date == 32) {
    tft.print('0');
    tft.setCursor(X_A2 + 62, Y_A2 + 159);
    newA2Date = 1;
    tft.print(newA2Date);
  }
  delay(t);
}

// if we press Date to DoW switch
if ((xpos >= X_A2 - 3) && (xpos <= X_A2 + 15) && (ypos >= Y_A2 + 157) && (ypos <= Y_A2 + 180)) {
  xpos = -1;
  ypos = -1;
  newA2Dy = !newA2Dy;
  newAlarmTwo = true;
  newHour2Selector = false;
  newMinute2Selector = false;
  tft.fillRect(X_A2 + 45, Y_A2 + 158, 40, 18, BLACK); // Draw a balck square over the last known value


  if (!newA2Dy) { // Draw selection of the day in a curent month

    // Turn on DoW selector
    newDoW2Selector = true;

    drawCheckMarkWhite(X_A2 + 13, Y_A2 + 159);
    tft.setTextColor(PINK);
```

```
      tft.setCursor(X_A2 + 50, Y_A2 + 159);
      if ((newA2Date <= 31) && (newA2Date >= 10)) {
        tft.print(newA2Date);
      }
      else if ((newA2Date < 10) && (newA2Date >= 1)) {
        tft.print('0');
        tft.setCursor(X_A2 + 62, Y_A2 + 159);
        tft.print(newA2Date);
      }
      tft.fillRect(X_A2 - 40, Y_A2 + 133, 18, 18, BLACK);
      tft.fillRect(X_A2 - 18, Y_A2 + 133, 18, 18, BLACK);
      tft.fillRect(X_A2 + 4, Y_A2 + 133, 18, 18, BLACK);
      tft.fillRect(X_A2 + 26, Y_A2 + 133, 18, 18, BLACK);
      tft.fillRect(X_A2 + 48, Y_A2 + 133, 18, 18, BLACK);
      tft.fillRect(X_A2 + 70, Y_A2 + 133, 18, 18, BLACK);
      tft.fillRect(X_A2 + 92, Y_A2 + 133, 18, 18, BLACK);
    }
    else {  // Draw selection of the week

      // Turn off DoW selector
      newDoW2Selector = false;

      // Draw check marks
      checkDoW2();

      // Draw black box over both areas and reset counters
      tft.fillRect(X_A2 + 12, Y_A2 + 158, 18, 18, BLACK);
    }
    delay(t);
  }

  // If we press set button
  if ((xpos >= X_A2 - 33) && (xpos <= X_A2 + 14) && (ypos >= Y_A2 + 182) && (ypos <= Y_A2 + 207)) {
    // Zero touchscreen
```

```
xpos = -1;
ypos = -1;
for (int i = rtc.getDoW(); i <= 7; i++) {
  if (i == 8) {
    i = i - 1;
  }
  if (alarmTwoWeek[i - 1] == true) {
    newA2Day = i;
    break;
  }
  else if (i == 7) {
    for (int i = 1; i <= 7; i++) {
      if (i == 8) {
        i = i - 1;
      }
      if (alarmTwoWeek[i - 1] == true) {
        newA2Day = i;
        break;
      }
    }
  }
}

if (newAlarmTwo) {
  newDoW2Selector = false;
  switch (newA2Day) {
    case 1:
      day_Of_The_Week = "Sunday";
      break;
    case 2:
      day_Of_The_Week = "Monday";
      break;
    case 3:
      day_Of_The_Week = "Tuesday";
      break;

    case 4:

      day_Of_The_Week = "Wednesday";

      break;

    case 5:

      day_Of_The_Week = "Thursday";

      break;

    case 6:

      day_Of_The_Week = "Friday";

      break;

    case 7:

      day_Of_The_Week = "Saturday";

      break;

  }

  tft.fillRect(X_A2 + 42, Y_A2 + 159, 45, 16, BLACK);

  tft.setCursor(X_A2 + 45, Y_A2 + 159);


  // Set Cursor, color and size

  tft.setTextSize(2);

  tft.setTextColor(PINK);


  if (newA2Dy) {

    String newA2SDay = day_Of_The_Week.substring(0, 3);

    tft.print(newA2SDay);

    // set Alarm

    if ((newA2h12) && (newA2PM) && ( newA2Hour <= 12)) newA2Hour = newA2Hour + 12;

    setAlarm(2, newA2Day, newA2Hour, newA2Minute, A1Second, A2Bits, newA2Dy, newA2h12 ,
newA2PM);

  }

  else {

    // Restore position

    tft.setCursor(X_A2 + 50, Y_A2 + 159);

    drawCheckMarkWhite(X_A2 + 13, Y_A2 + 159);

    newDoW2Selector = true;
```

```
          if ((newA2Date <= 31) && (newA2Date >= 10)) {
            tft.print(newA2Date);
          }
          else if ((newA2Date < 10) && (newA2Date >= 1)) {
            tft.print('0');
            tft.setCursor(X_A2 + 62, Y_A2 + 159);
            tft.print(newA2Date);
          }

          // Set alarm
          if ((newA2h12) && (newA2PM) && ( newA2Hour <= 12)) newA2Hour = newA2Hour + 12;
          setAlarm(2, newA2Date, newA2Hour, newA2Minute, A1Second, A2Bits, newA2Dy, newA2h12 ,
newA2PM);
        }

        // Change Status
        rtc.turnOnAlarm(2); // Turn alarm on
        tft.setCursor(X_A2 + 65, Y_A2 + 51);
        tft.fillRect(X_A2 + 65, Y_A2 + 51, 34, 14, BLACK);
        checkAlarmStatus(2);

      }
    }

    if (newA2Dy) { // Are we setting up alarm for the days of the week? operate checkmarks. If false than
we setting it up for the date in a month

      // If we click in the first check Box
      if ((xpos >= X_A2 - 50) && (xpos <= X_A2 - 34) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 154)) {
        newAlarmTwo = true;
        if (alarmTwoWeek[0] == false) {
          // Zero touchscreen
          xpos = -1;
          ypos = -1;
          drawCheckMarkRed(X_A2 - 39, Y_A2 + 134);
          alarmTwoWeek[0] = true;
          delay(t);
        } else {
          // Zero touchscreen
          xpos = -1;
          ypos = -1;
          tft.fillRect(X_A2 - 40, Y_A2 + 133, 18, 18, BLACK);
          alarmTwoWeek[0] = false;
          delay(t);
        }
      }

      // If we click in the second check Box
      if ((xpos >= X_A2 - 33) && (xpos <= X_A2 - 17) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 152)) {
        newAlarmTwo = true;
        // If button is on or off
        if (alarmTwoWeek[1] == false) {
          // Zero touchscreen
          xpos = -1;
          ypos = -1;
          drawCheckMark(X_A2 - 17, Y_A2 + 134);
          alarmTwoWeek[1] = true;
          delay(t);
        } else {
          // Zero touchscreen
          xpos = -1;
          ypos = -1;
          tft.fillRect(X_A2 - 18, Y_A2 + 133, 18, 18, BLACK);
          alarmTwoWeek[1] = false;
          delay(t);
        }
      }
```

```
// If we click in the check Box #3
if ((xpos >= X_A2 - 11) && (xpos <= X_A2 + 7) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 152)) {
  newAlarmTwo = true;
  // If button is on or off
  if (alarmTwoWeek[2] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A2 + 5, Y_A2 + 134);
    alarmTwoWeek[2] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A2 + 4, Y_A2 + 133, 18, 18, BLACK);
    alarmTwoWeek[2] = false;
    delay(t);
  }
}

// If we click in the check Box #4
if ((xpos >= X_A2 + 11) && (xpos <= X_A2 + 29) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 152)) {
  newAlarmTwo = true;
  // If button is on or off
  if (alarmTwoWeek[3] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A2 + 27, Y_A2 + 134);
    alarmTwoWeek[3] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A2 + 26, Y_A2 + 133, 18, 18, BLACK);
    alarmTwoWeek[3] = false;
    delay(t);
  }
}

// If we click in the check Box #5
if ((xpos >= X_A2 + 33) && (xpos <= X_A2 + 51) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 152)) {
  newAlarmTwo = true;
  // If button is on or off
  if (alarmTwoWeek[4] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A2 + 49, Y_A2 + 134);
    alarmTwoWeek[4] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A2 + 48, Y_A2 + 133, 18, 18, BLACK);
    alarmTwoWeek[4] = false;
    delay(t);
  }
}

// If we click in the check Box #6
if ((xpos >= X_A2 + 55) && (xpos <= X_A2 + 73) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 152)) {
  newAlarmTwo = true;
  // If button is on or off
  if (alarmTwoWeek[5] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMark(X_A2 + 71, Y_A2 + 134);
    alarmTwoWeek[5] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A2 + 70, Y_A2 + 133, 18, 18, BLACK);
    alarmTwoWeek[5] = false;
    delay(t);
  }
}

// If we click in the check Box #7
if ((xpos >= X_A2 + 77) && (xpos <= X_A2 + 95) && (ypos >= Y_A2 + 132) && (ypos <= Y_A2 + 152)) {
  newAlarmTwo = true;
  // If button is on or off
  if (alarmTwoWeek[6] == false) {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    drawCheckMarkRed(X_A2 + 93, Y_A2 + 134);
    alarmTwoWeek[6] = true;
    delay(t);
  } else {
    // Zero touchscreen
    xpos = -1;
    ypos = -1;
    tft.fillRect(X_A2 + 92, Y_A2 + 133, 18, 18, BLACK);
    alarmTwoWeek[6] = false;
```

```
      delay(t);
    }
  }

}

// If we press clear button
if ((xpos >= X_A2 + 24) && (xpos <= X_A2 + 74) && (ypos >= Y_A2 + 182) && (ypos <= Y_A2 + 207)) {
  // Zero touchscreen
  xpos = -1;
  ypos = -1;
  rtc.turnOffAlarm(2);
  tft.setCursor(X_A2 + 65, Y_A2 + 51);
  tft.fillRect(X_A2 + 65, Y_A2 + 51, 34, 14, BLACK);
  checkAlarmStatus(2);

  // Draw one instead of current alarm 1 hour
  newA2Hour = 1;
  newAlarmTwo = false;

  if (!A2h12) {
    tft.setCursor(X_A2, Y_A2 + 94);
  }  else tft.setCursor(X_A2 - 11, Y_A2 + 94);
  tft.setTextColor(BLUE);
  tft.setTextSize(4);
  if (!A2h12) {
    tft.fillRect(X_A2, Y_A2 + 94, 44, 28, BLACK);
  } else tft.fillRect(X_A2 - 11, Y_A2 + 94, 44, 28, BLACK);
  tft.print('0');
  if (!A2h12) {
    tft.setCursor(X_A2 + 24, Y_A2 + 94);
  } else tft.setCursor(X_A2 + 13, Y_A2 + 94);
  tft.print(newA2Hour);
```

```
// Draw one instead of current alarm 1 minute
newA2Minute = 1;
if (!A2h12) {
  tft.fillRect(X_A2 + 55, Y_A2 + 94, 44, 28, BLACK);
  tft.setCursor(X_A2 + 55, Y_A2 + 94);
} else {
  tft.fillRect(X_A2 + 43, Y_A2 + 94, 44, 28, BLACK);
  tft.setCursor(X_A2 + 43, Y_A2 + 94);
}
tft.print('0');
if (!A2h12) {
  tft.setCursor(X_A2 + 79, Y_A2 + 94);
} else tft.setCursor(X_A2 + 67, Y_A2 + 94);
tft.print(newA2Minute);

// Turn off existing check marks
tft.fillRect(X_A2 - 40, Y_A2 + 133, 18, 18, BLACK);
tft.fillRect(X_A2 - 18, Y_A2 + 133, 18, 18, BLACK);
tft.fillRect(X_A2 + 4, Y_A2 + 133, 18, 18, BLACK);
tft.fillRect(X_A2 + 26, Y_A2 + 133, 18, 18, BLACK);
tft.fillRect(X_A2 + 48, Y_A2 + 133, 18, 18, BLACK);
tft.fillRect(X_A2 + 70, Y_A2 + 133, 18, 18, BLACK);
tft.fillRect(X_A2 + 92, Y_A2 + 133, 18, 18, BLACK);

// Draw black box over both areas and reset counters
tft.fillRect(X_A2 + 12, Y_A2 + 158, 18, 18, BLACK);
tft.fillRect(X_A2 + 45, Y_A2 + 158, 40, 18, BLACK);
newA2Date = 1;
newA2Day = 1;

newA2Dy = true;
for (int i = 0; i <= 6; i++) {
  alarmTwoWeek[i] = false;
  EEPROM.write(eeAddressAlarmTwo + i, alarmTwoWeek[i]);
```

```
    }
  }

} // End of page 3

// Radio screen
if (currentPage == 4) {

  // Draw date/time/temp/day of the week
  drawTemp();
  drawDate();
  drawDayOfTheWeek();
  drawSmallClock();
  touch_Screen_Read();

  // If we pressing back button
  if ((ypos > tft.height() - 40) && (xpos < 40)) {
    zeroAllData();
    currentPage = 2;
    draw_Media_Screen();
  }

} // End of a Radio screen

// MP3 Player sscreen
if (currentPage == 5) {
  drawTemp();
  drawDate();
  drawDayOfTheWeek();
  drawSmallClock();
  touch_Screen_Read();
  if (playBackStatus == 0) {
    drawMediaButton(208, 128);
  }
```

```
else if (playBackStatus == 1) {
  drawPause(211, 130);
}

// If we pres play or pause
if ((xpos > 194) && (xpos < 270) && (ypos > 128) && (ypos < 190)) {
  xpos = -1;
  ypos = -1;
  tft.fillRect(209, 129, 63, 62, BLACK);
  if (playBackStatus == 0) {
    mp3.play();
    playBackStatus = 1;
  } else if (playBackStatus == 1) {
    mp3.pause();
    playBackStatus = 0;
  }
  delay(t * 2);
}

// If we press volume Up
if ((xpos > 410) && (xpos < 470) && (ypos > 150) && (ypos < 175)) {
  xpos = -1;
  ypos = -1;
  if (vol < 30) {
    ++vol;
  }
  mp3.setVolume(vol);
}

// If we press volume Down
if ((xpos > 10) && (xpos < 40) && (ypos > 150) && (ypos < 175)) {
  xpos = -1;
  ypos = -1;
  if (vol > 0) {
    --vol;
  }
  mp3.setVolume(vol);
}

// If we press Previus button
if ((xpos > 70) && (xpos < 150) && (ypos > 150) && (ypos < 175)) {
  mp3.previousTrack();
  xpos = -1;
  ypos = -1;
  tft.fillRect(209, 129, 63, 62, BLACK);
  drawPause(211, 130);
  playBackStatus = 1;
  delay(t * 2);
}

// If we press Next button
if ((xpos > 320) && (xpos < 390) && (ypos > 150) && (ypos < 175)) {
  mp3.nextTrack();
  xpos = -1;
  ypos = -1;
  tft.fillRect(209, 129, 63, 62, BLACK);
  drawPause(211, 130);
  playBackStatus = 1;
  delay(t * 2);
}

// If we press back button
if ((ypos > tft.height() - 40) && (xpos < 40)) { // If we pressing back button
  xpos = -1;
  ypos = -1;
  zeroAllData();
  draw_Media_Screen();
  currentPage = 2;
  }

} // End of MP3 loop

// Bluetooth Screen
if (currentPage == 6) {
  drawTemp();
  drawDate();
  drawDayOfTheWeek();
  drawSmallClock();
  touch_Screen_Read();
  if ((ypos > tft.height() - 40) && (xpos < 40)) { // If we pressing back button
    xpos = -1;
    ypos = -1;
    zeroAllData();
    draw_Media_Screen();
    currentPage = 2;
  }
} // End of the blutotth screen

// If Tilt switch lights up Red it has value of 0
if (digitalRead(tiltSwitch) == NULL) {
  mp3.pause();
  playBackStatus = 0;

  if (rtc.checkAlarmEnabled(1)) { // if Alarm 1 was on
    rtc.turnOffAlarm(1);
    delay(t);
    rtc.turnOnAlarm(1);
  }
  if (rtc.checkAlarmEnabled(2)) { // If alarm two was on
    rtc.turnOffAlarm(2);
    delay(t);
    rtc.turnOnAlarm(2);
    }

    if (currentPage == 5) { // If in a Mp3 player screen
      tft.fillRect(209, 129, 63, 62, BLACK);
      drawMediaButton(208, 128);
    }
  }

  // Set Alarm for the next day in current week
  resetAlarmWhenDoW();

  // Play music if Alarm 1 or 2 is ON and Ringing
  if((newAlarmTwo) || (newAlarmOne)){
   activateMusicIfAlarm(); // Call also clear the flag in RTC library
  }
} // End of the void loop

[/code]
```