Smart Watch

S.T.E.M Kit

What is STEM?

STEM stands for Science, Technology, Engineering and Mathematics

The aim of STEM is the integration of these four disciplines together in teaching and learning. As, in the real world, these four disciplines rely heavily and seamlessly on each other.

STEM helps strengthen key life skills such as analytical thinking, problem solving, creativity, teamwork and technical skills

What is WiTT?

WiTT stands for Women in Technology and Trades

WiTT is a group that increases opportunities and support for women in technology and trades in all fields, through a rich networking and support community

WiTT welcomes industry, staff, students and faculty across all areas of the college and all genders, backgrounds, races and orientation to become involved and contribute to the support of women in technology and/or trades.

WiTT STEM Kit
Smart Watch Kit

This kit will give students a hands-on experience with STEM, with a focus on the Technology and Engineering aspects. With this kit, students will learn:

- What each hardware component is, and how they work.
- How 3D printing works and using software to create their own watch case.
- How to code basic watch functions using Arduino IDE.
- How to construct and solder circuits.

From coding watch functions to assembling the watch, students will receive firsthand experience with STEM. Students will come out with their very own basic smart watch that can be continuously experimented with to bring more functionality!

Curriculum Points

Grade 10, Open. Page 58. Part A. Computer Technology Fundamentals. Subpart A1. Computer Hardware. Section A.1.3 Identifying the basic components and peripheral devices of a computer system.

Grade 10, Open. Page 59. Part B. Computer Technology Skills. Subpart B1. Workstation Setup. Section B1.3 Using file-management techniques to organize and back up files. Subpart B2. Electronics, Robotics, and Computer Interfacing. Section B2.1 Safely construct and test electronic circuits using both breadboard and soldering techniques to connect discrete components and/or integrated circuits. Subpart B4. Software. Section B4.2 Installing and configuring software on a workstation.

Grade 10, Open. Page 60. Part B. Computer Technology Skills. Subpart B5. Computer Programming. Section B5.1 Using a procedural programming language to define constants and variables, write expressions and assignment statements, and specify the order in which the operations are performed in a program. Section B5.2 Using input and output statements in a program.

Grade 10, Open. Page 36. Part B. Introduction to programming. Subpart B1. Programming concepts. Section B1.5 Identifying situations in which decision and looping structures are required. Subpart B2. Writing Programs. Section B2.5 writing programs that use looping structures effectively.

Grade 11, University Preparation. Page 40. Part A. Programming Concepts and Skills. Subpart A3. Subprograms. Section A3.2 Writing subprograms that use parameter passing and appropriate variable scope, to perform tasks within programs.

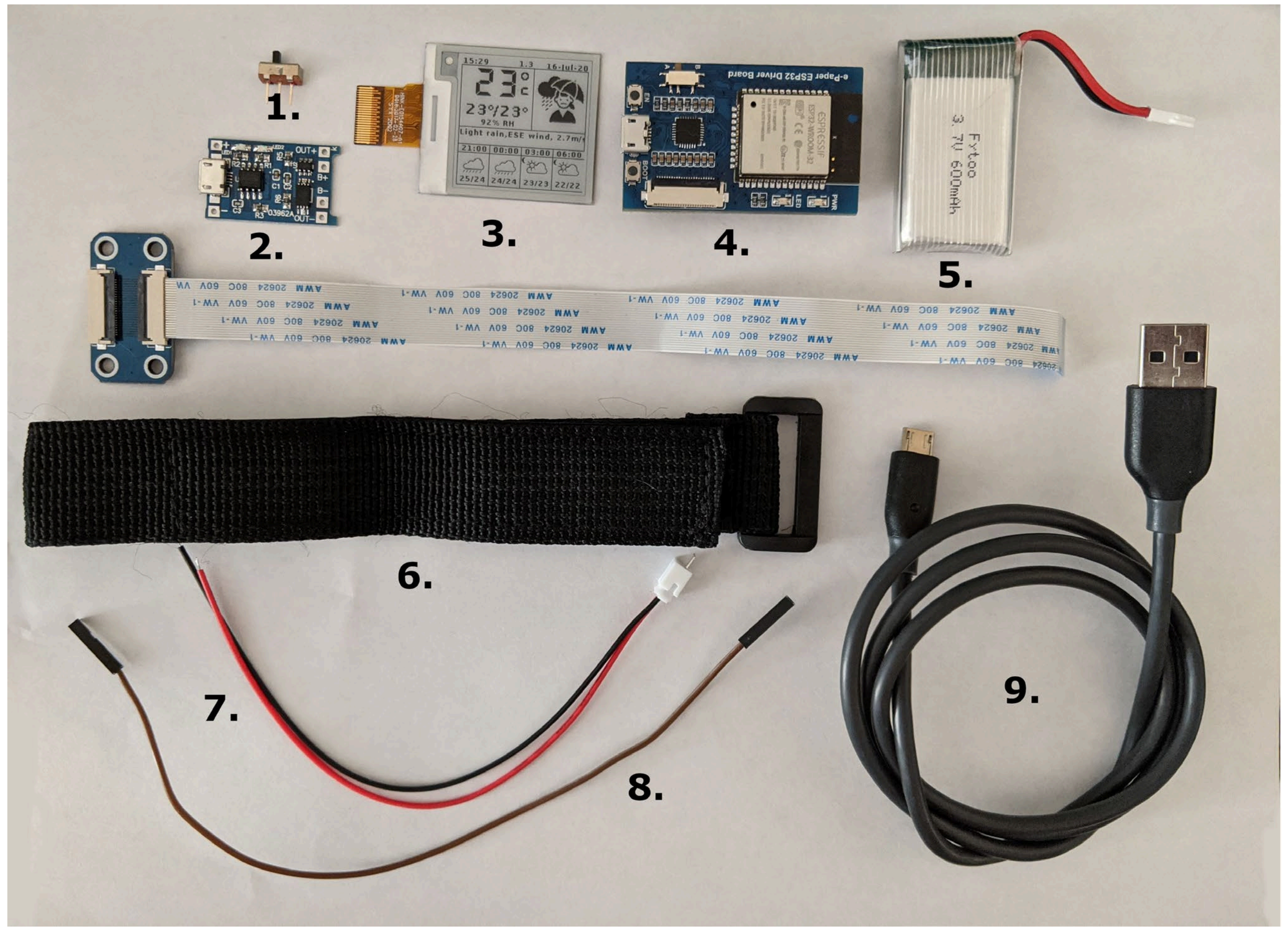
Grade 11, College Preparation. Page 48. Part A. Programming Concepts and Skills. Subpart A1. Data types and Expressions. Section A1.2 Demonstrate the ability to manipulate string data in a computer program. Section A1.3 Using assignment statements correctly with both arithmetic and string expressions in computer programs.

Grade 12, University Preparation. Page 56. Part A. Programming Concepts and Skills. Subpart A3. Designing algorithms. Section A3.1 Demonstrate the ability to read from, and write to, an external file.

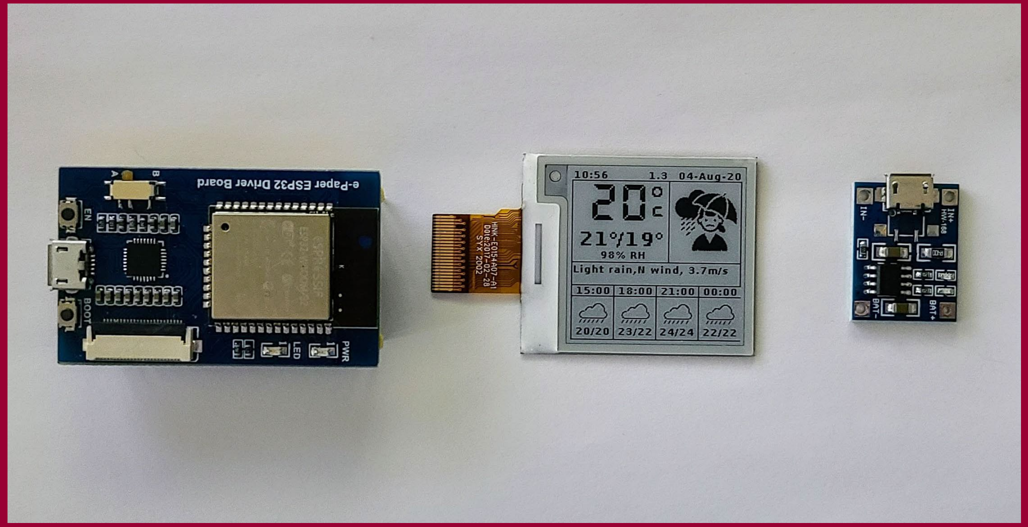
Grade 12, College Preparation. Page 66. Part B. Software Development. Subpart B3. Graphical user interfaces. Section B3.1 Designing graphical user interfaces that contain common controls.

Smart Watch Kit
Kit Requirements

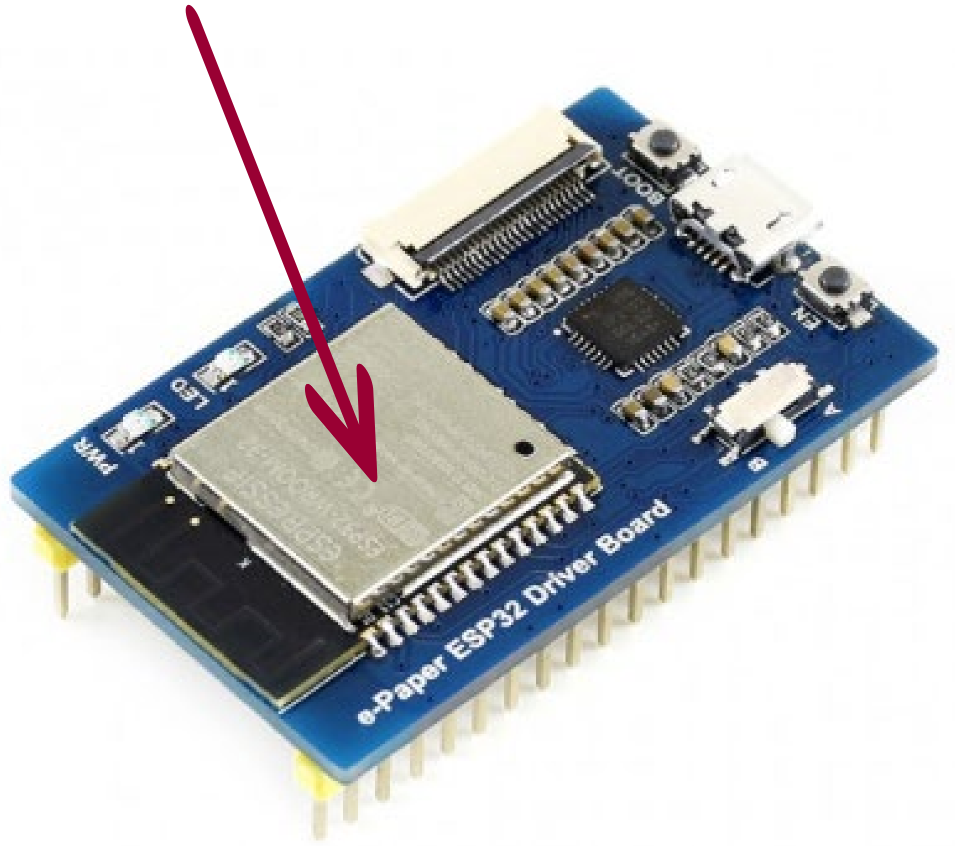
1. Small SPDT Switch
2. TP4056 Charging Board
3. Waveshare 1.54inch E-Ink Raw Display Panel
4. Waveshare e-Paper ESP32 Driver Board
5. 3.7V 600mah Lithium-ion Polymer Battery
6. Velcro Wrist Strap
7. Mini Micro Jst 2.0 Ph 2-Pin Connector Male Plug
8. Jumper Wires
9. Micro USB to USB Cable
10. 3D Printer
11. Soldering Iron
12. Arduino IDE



HARDWARE


e-Paper ESP32 Driver Board
10:56
04-Aug-20
20°C
21°/19°
98% RH
Light rain,N wind, 3.7m/s
15:00
18:00
21:00
00:00
20/20
23/22
24/24
22/22

Microcontroller Board



1.

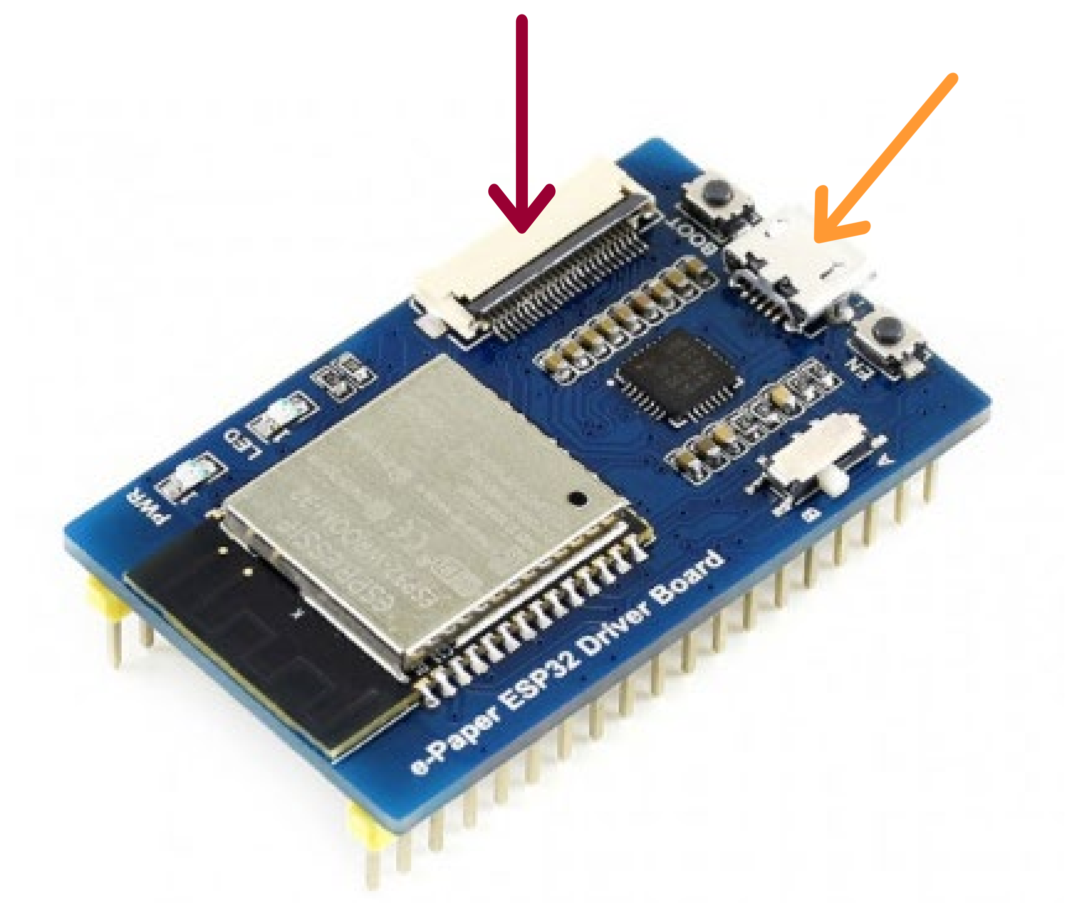
Microcontroller

A microcontroller is a compact integrated circuit designed to control a specific operation in an **embedded system** (a combination of computer hardware and software designed for a specific function or functions). A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

2.

3.

Microcontroller Board



1.

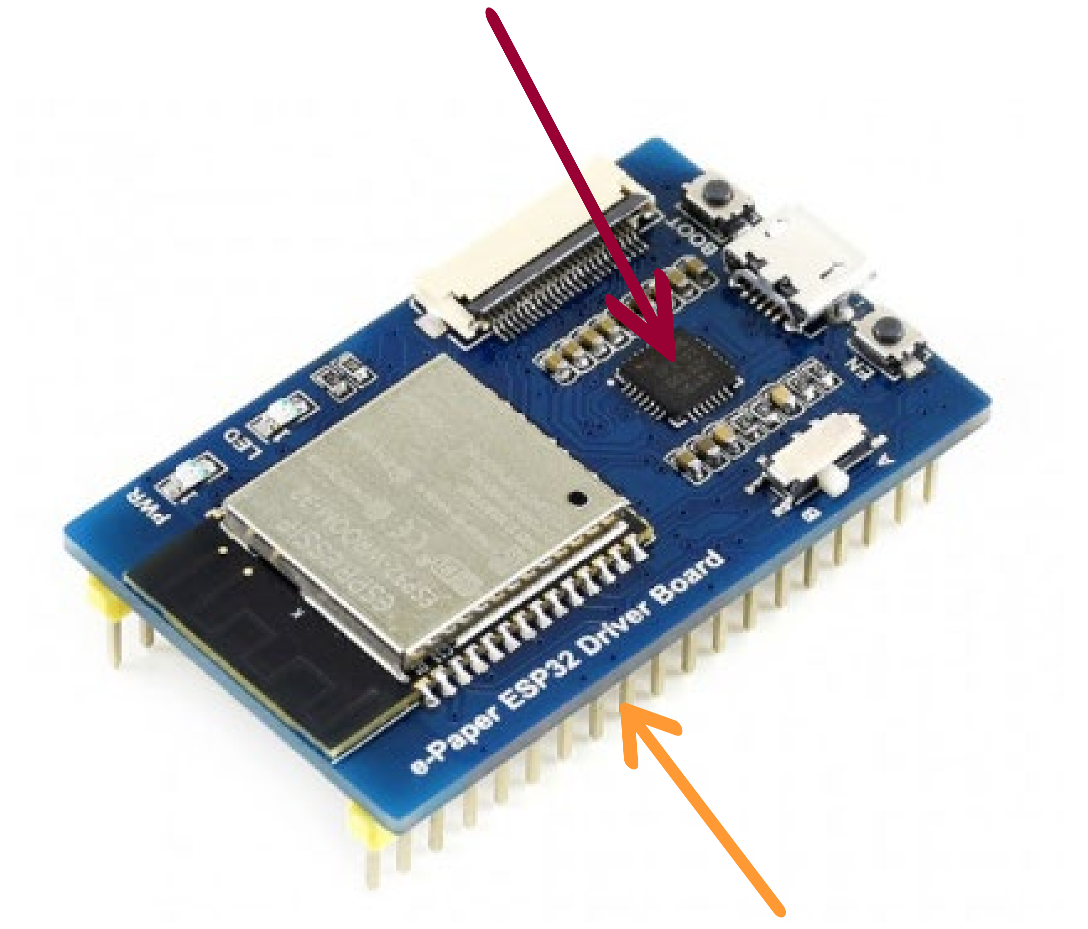
SPI

A Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and peripherals. 4.

Micro USB Port

A micro Universal Serial Bus (USB) port is a standard cable connection interface for personal computers and consumer electronics devices. 5.

Microcontroller Board



1.

USB to UART Bridge

A Universal Serial Bus (USB) to Universal Asynchronous Receiver/Transmitter (UART) allows UBS connectivity to a device with a UART interface such as a microcontroller. 6.

I/O Connectors

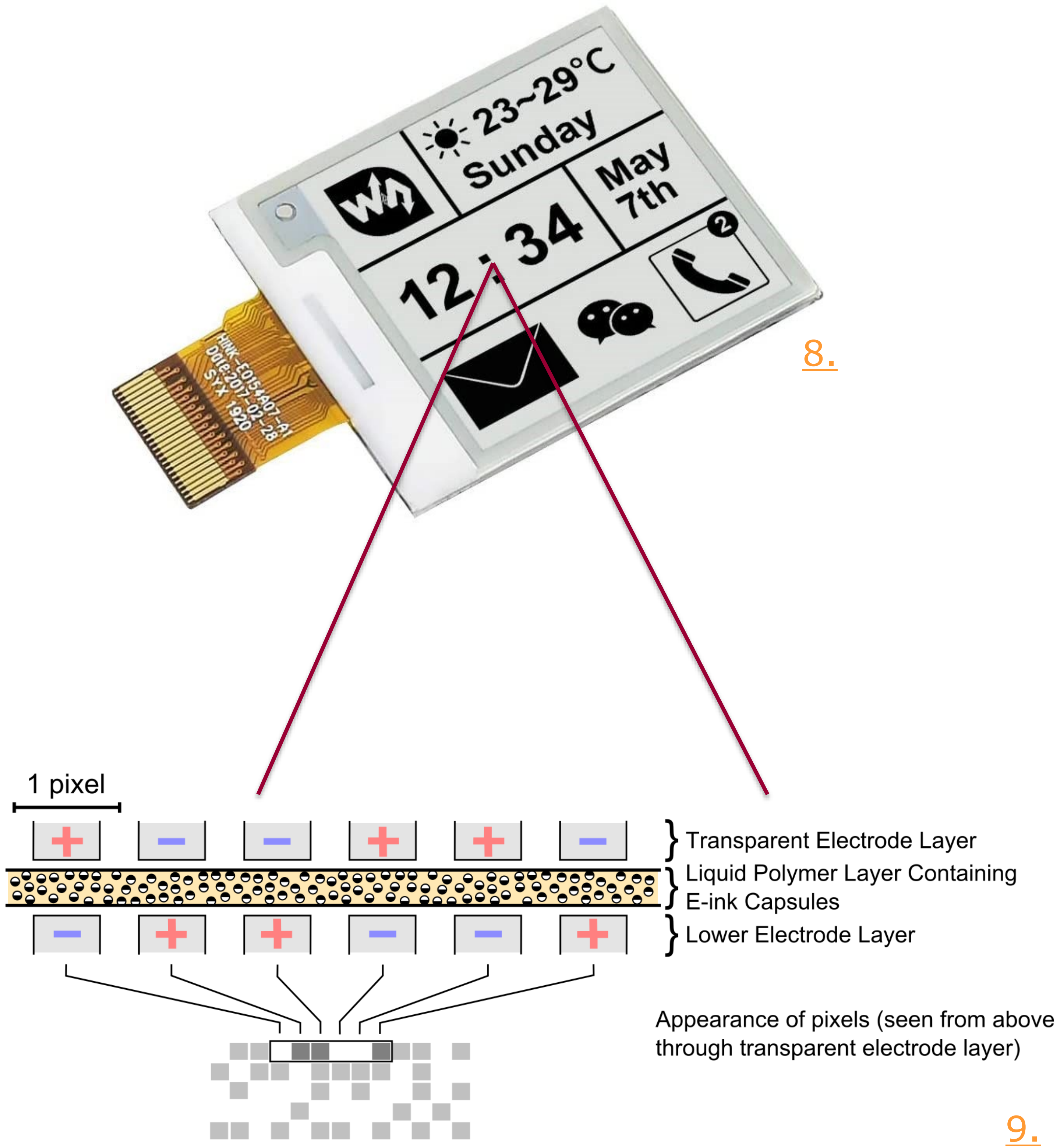
Input/output connectors are an electrical contact that allows data transmission to external devices. 7.

Display Panel

E-Ink Display

Electronic Ink displays are used to mimic the appearance and reflect light like pen on paper.

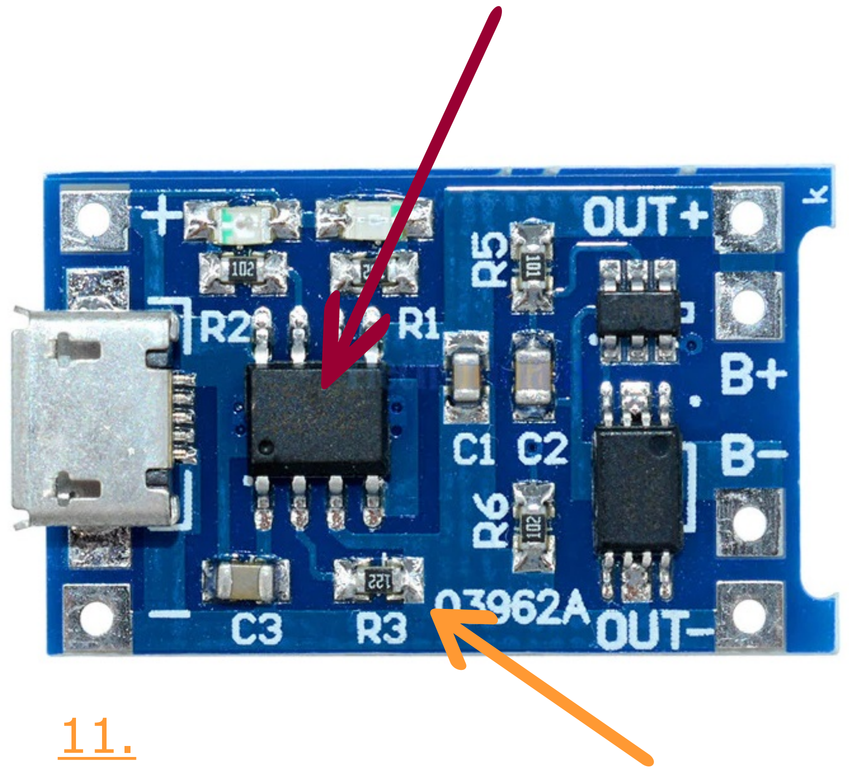
These displays work by taking thousands of capsules only 100 microns wide filled with pigment chips and suspending them in a liquid polymer. When an electronic charge is applied to the capsules, the pigment chips will either be pushed to the top where they appear white or pulled to the bottom where they appear black. 10.


23~29°C
Sunday
May 7th
12:34
1 pixel
Transparent Electrode Layer
Liquid Polymer Layer Containing E-ink Capsules
Lower Electrode Layer
Appearance of pixels (seen from above through transparent electrode layer)

8.

9.

Charging Board



11.

TP4056 Chip

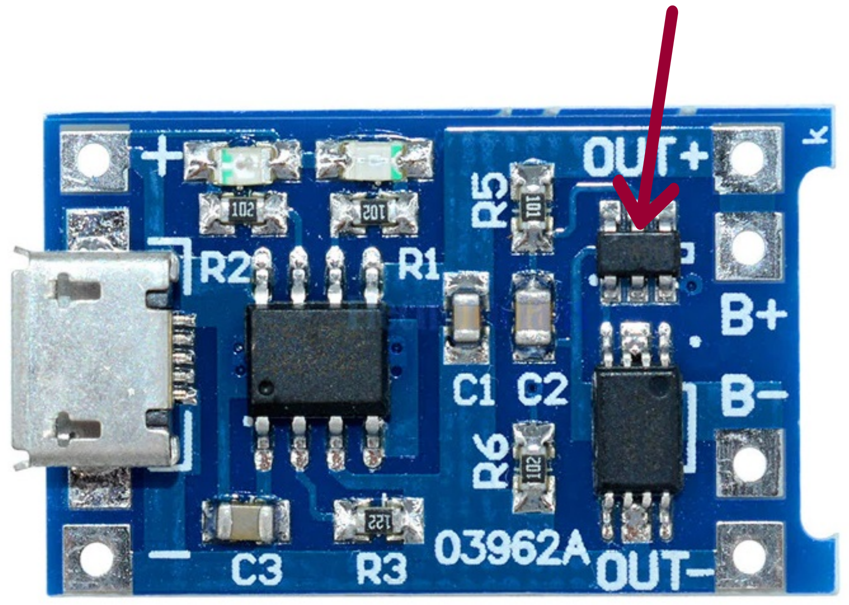
The TP4056 is a chip used to charge single cell lithium-ion polymer battery. This chip allows the battery to charge at a constant current/voltage, with built-in protections to stop the battery from over charging (which would damage it), as well as a trickle charge threshold which prevents the battery from outputting a low voltage while in use that could damage the board it is powering.

Rprog

The programming resistor (Rprog/R3) is a resistor set to 1200 ohms which allows the battery to charge at a 1-amp charge rate.

12.

Charging Board

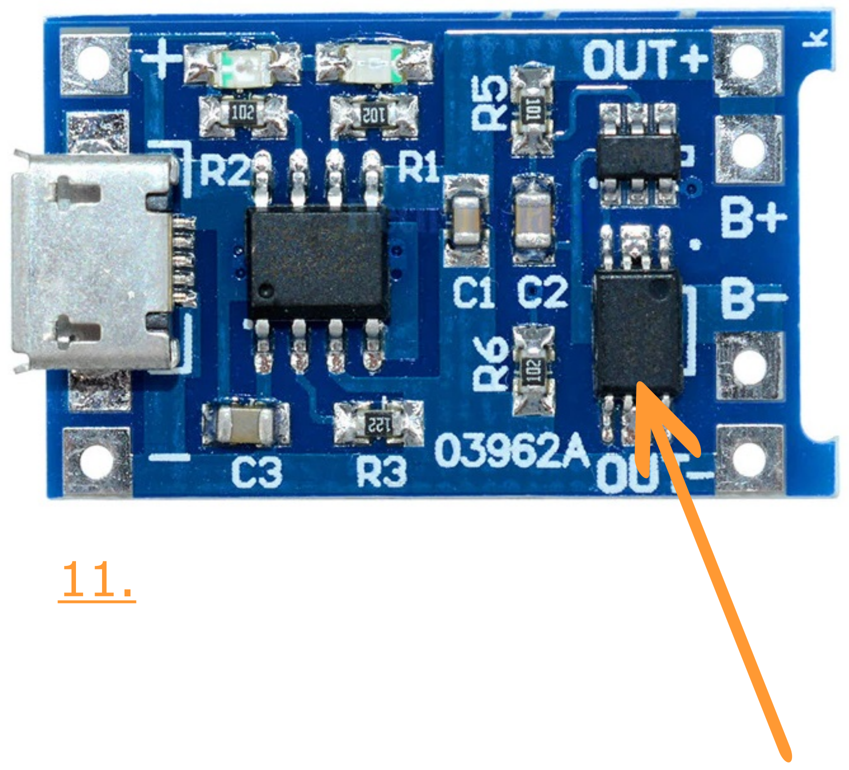


11.

DW01A Protector Chip

The DW01A chip provides short circuit protection by monitoring the charger input with short circuit, over current, charger, and reverse charger detectors. It also monitors the battery with an overcharge detector (battery voltage is too high) and overdischarge detector (battery voltage is too low). 12.

Charging Board



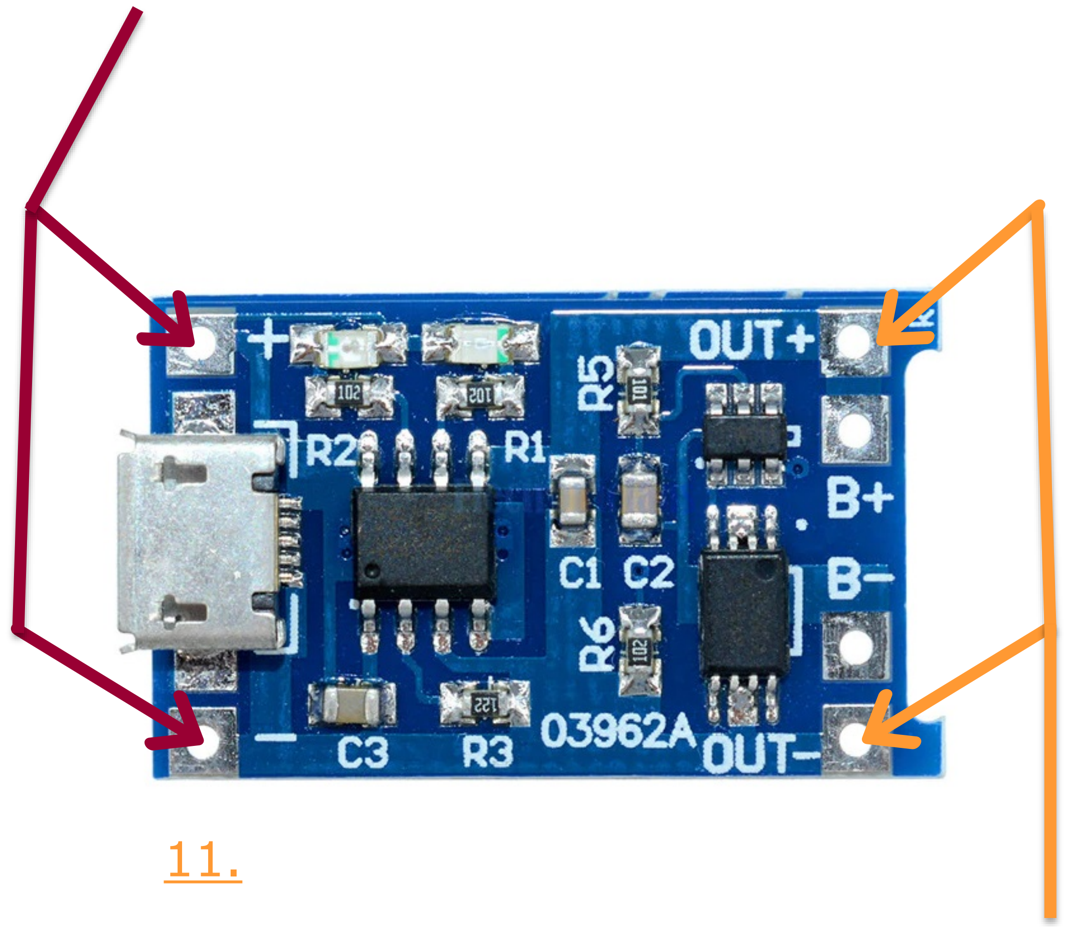
11.

8205A Duel MOSFET

A duel Metal Oxide Semiconductor Field Effect Transistor (MOFSET) is a chip made up of 2 MOSFETs in series. These transistors are made up by 3 pins – the gate, the drain, and the source that work as a variable resistor controlled by voltage. When voltage is applied to the gate and the source, current can flow from the drain to the source. The current is variable depending on the voltage applied to the gate. In this case the duel MOSFET is used to limit the current flowing into the battery, stopping the battery from charging once it reaches its max voltage.

13.

Charging Board



11.

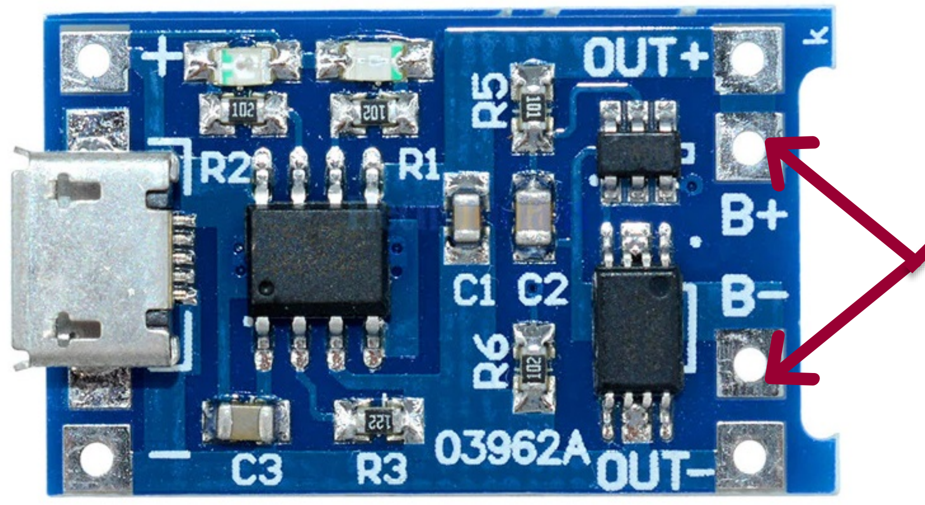
Voltage In Plated-Through Holes

Voltage in plated-through holes allows you to connect to another voltage source to charge the battery instead of using the micro USB port. The plus end connects to the positive end of the voltage source, while the negative connected to the ground.

Voltage Out Plated-Through Holes

Voltage out plated-through holes allows you to power a component with the lithium-ion polymer battery. The plus end is connected to the positive end of the voltage in on the component, while negative end is connected to the ground. 12.

Charging Board



11.

Battery Plated-Through Holes

The battery plated-through holes are the points where the board connects to the battery. The plus connects to the positive end of the battery, while negative connects to the negative end of the battery.

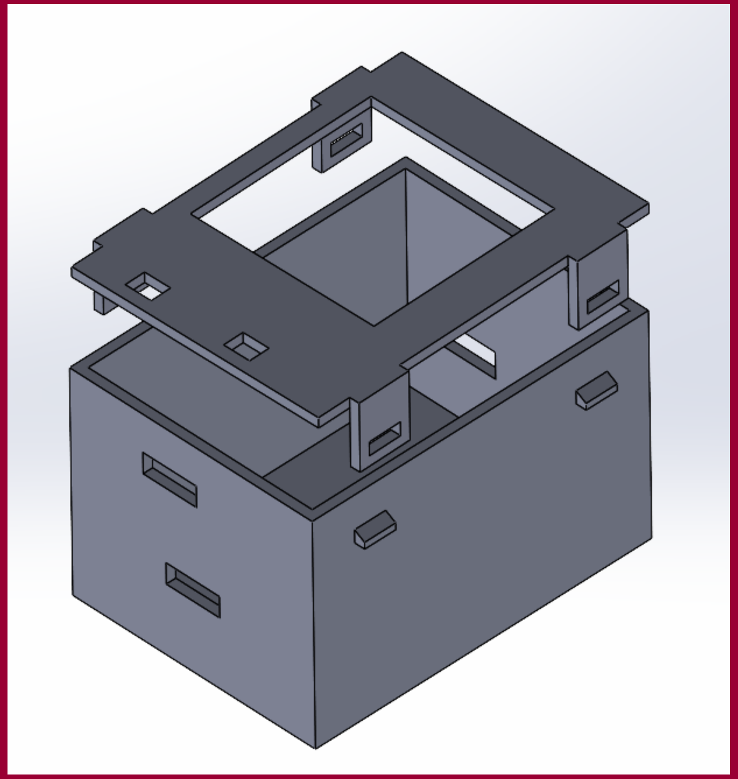
12.

Time to Reflect!

1. What other hardware components do you think you could add?

2. What are some electronics that would use these components?

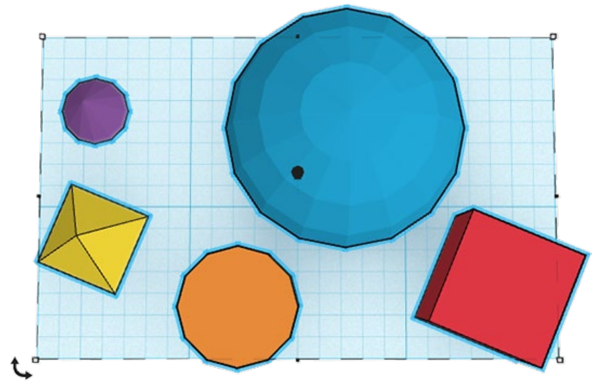
3D PRINTING



Computer Aided Design (CAD)

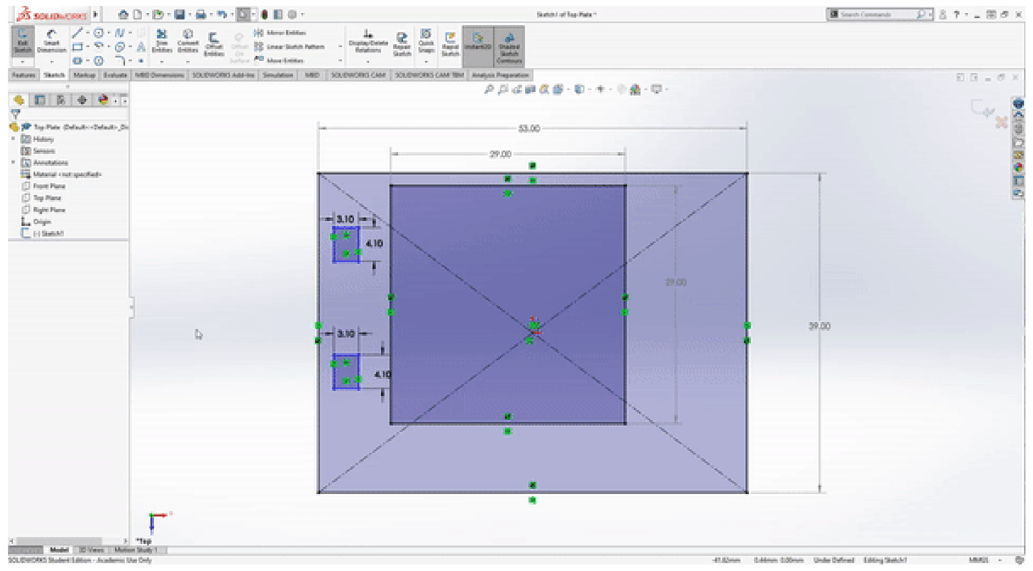
3D Modeling

Models are created in a 3-dimensional space where they are made up of 2-dimensional surfaces known as a "plane".



Great for beginners! → AUTODESK TINKERCAD

Various shapes can be sketched on these planes for example lines, circles, squares, and ellipses.



These sketches can be modified by doing actions such as extruded, cut out, revolved, lofted, and many more.

Wide range of free software available for students, ranging in levels of expertise.

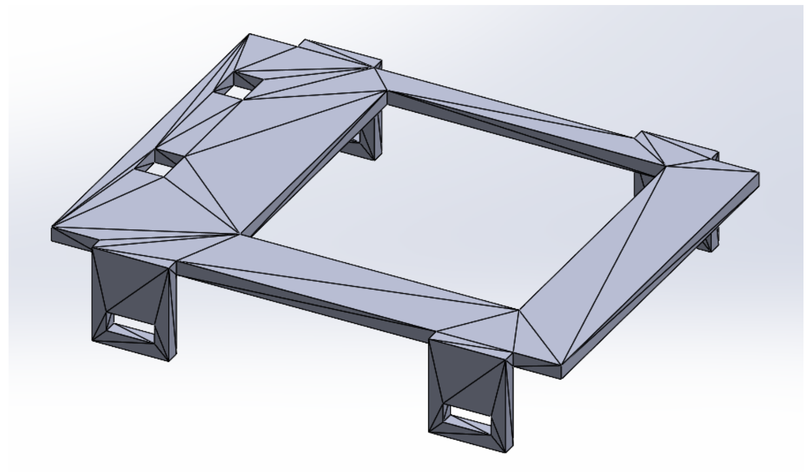
blender
AUTODESK FUSION 360
SOLIDWORKS | Apps for Kids
AUTODESK TINKERCAD
SketchUp
SOLIDWORKS
AUTOCAD
FreeCAD
AUTODESK INVENTOR

14,15.

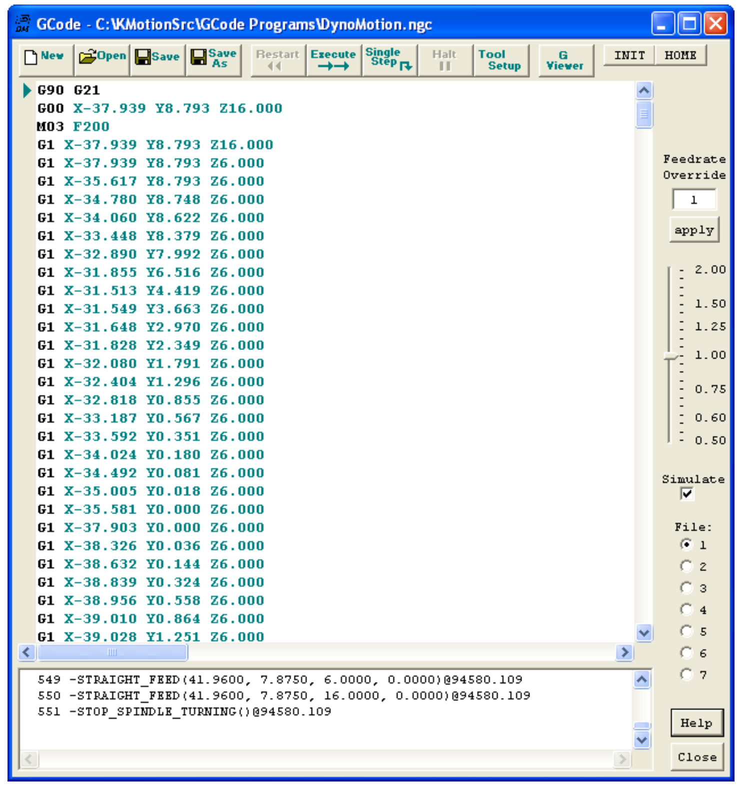
Slicing and File Formats

First the 3D model will be saved as a 3D object file.

This object must be "sliced" before it can be printed, and the slicer software requires a specific format known as Standard Triangle Language (.STL)



"Slicing" is a computer-generated process that takes an .STL model and converts it into code that the 3D printer can read; .GCODE

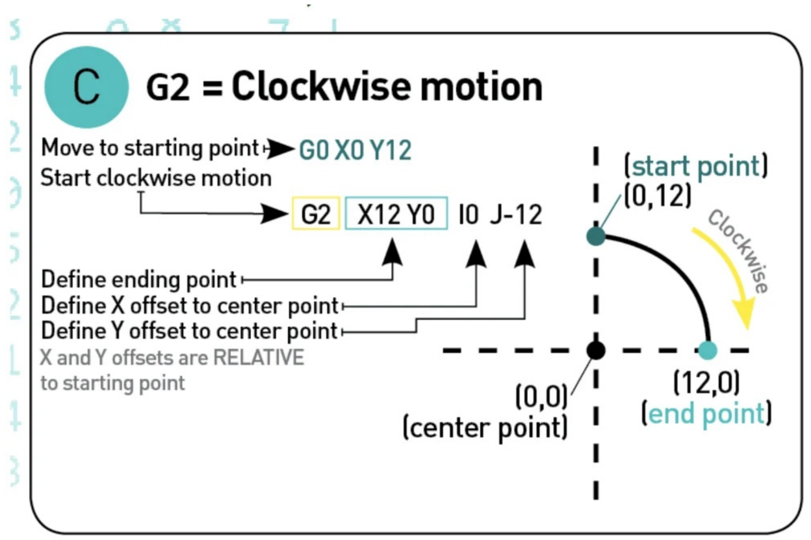


```
G90 G21
G00 X-37.939 Y8.793 Z16.000
M03 F200
G1 X-37.939 Y8.793 Z16.000
G1 X-37.939 Y8.793 Z6.000
G1 X-35.617 Y8.793 Z6.000
G1 X-34.780 Y8.748 Z6.000
G1 X-34.060 Y8.622 Z6.000
G1 X-33.448 Y8.379 Z6.000
G1 X-32.890 Y7.992 Z6.000
G1 X-31.855 Y6.516 Z6.000
G1 X-31.513 Y4.419 Z6.000
G1 X-31.549 Y3.663 Z6.000
G1 X-31.648 Y2.970 Z6.000
G1 X-31.828 Y2.349 Z6.000
G1 X-32.080 Y1.791 Z6.000
G1 X-32.404 Y1.296 Z6.000
G1 X-32.818 Y0.855 Z6.000
G1 X-33.187 Y0.567 Z6.000
G1 X-33.592 Y0.351 Z6.000
G1 X-34.024 Y0.180 Z6.000
G1 X-34.492 Y0.081 Z6.000
G1 X-35.005 Y0.018 Z6.000
G1 X-35.581 Y0.000 Z6.000
G1 X-37.903 Y0.000 Z6.000
G1 X-38.326 Y0.036 Z6.000
G1 X-38.632 Y0.144 Z6.000
G1 X-38.839 Y0.324 Z6.000
G1 X-38.956 Y0.558 Z6.000
G1 X-39.010 Y0.864 Z6.000
G1 X-39.028 Y1.251 Z6.000
```

```
549 -STRAIGHT_FEED(41.9600, 7.8750, 6.0000, 0.0000)@94580.109
550 -STRAIGHT_FEED(41.9600, 7.8750, 16.0000, 0.0000)@94580.109
551 -STOP_SPINDLE_TURNING()@94580.109
```

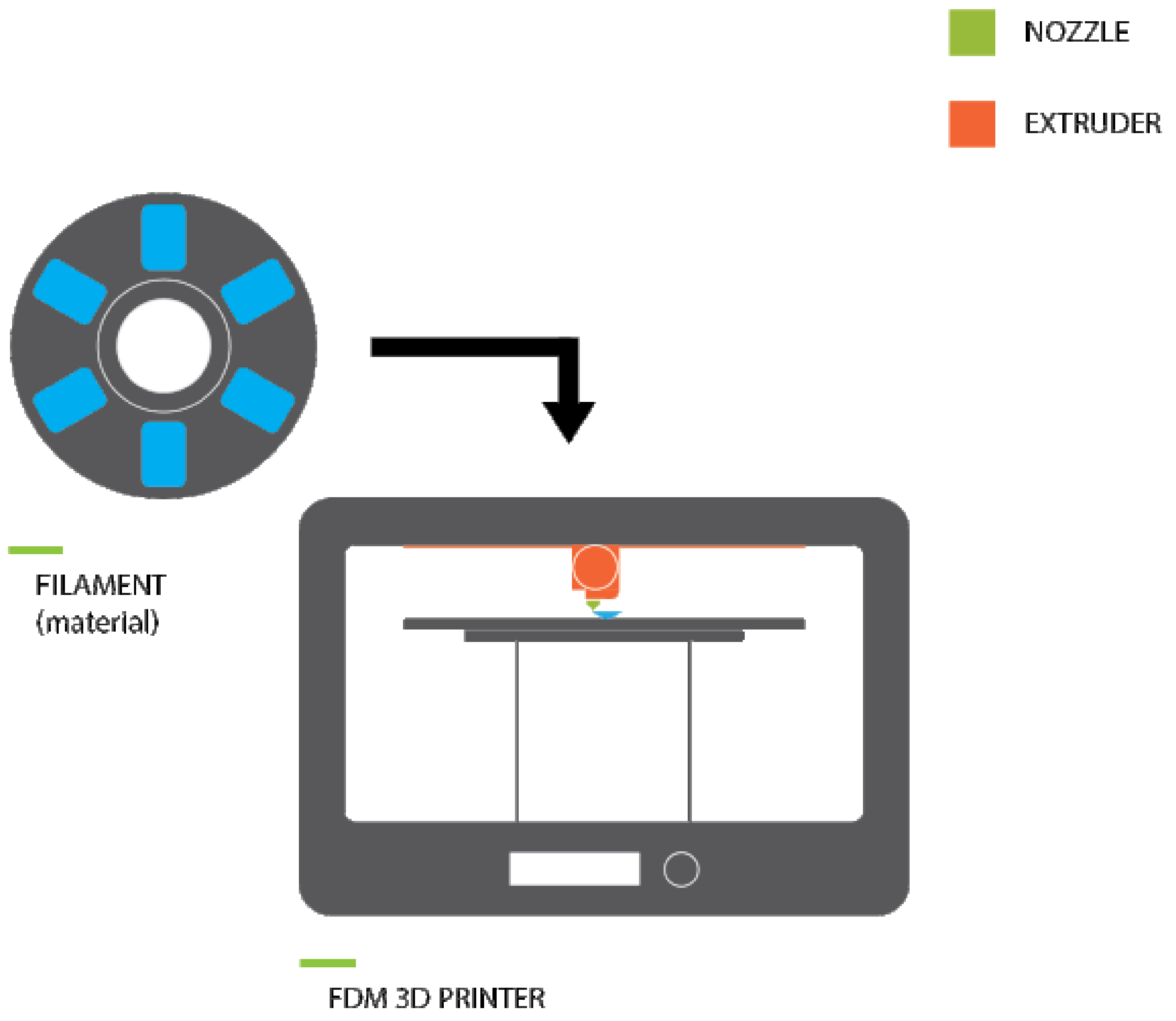
16.

.GCODE breaks the file down into individual layers and movements for the printer to read. This code can be hundreds of pages long for complex prints!



17.

Fused Deposition Modeling (FDM)



18.

3D Printing

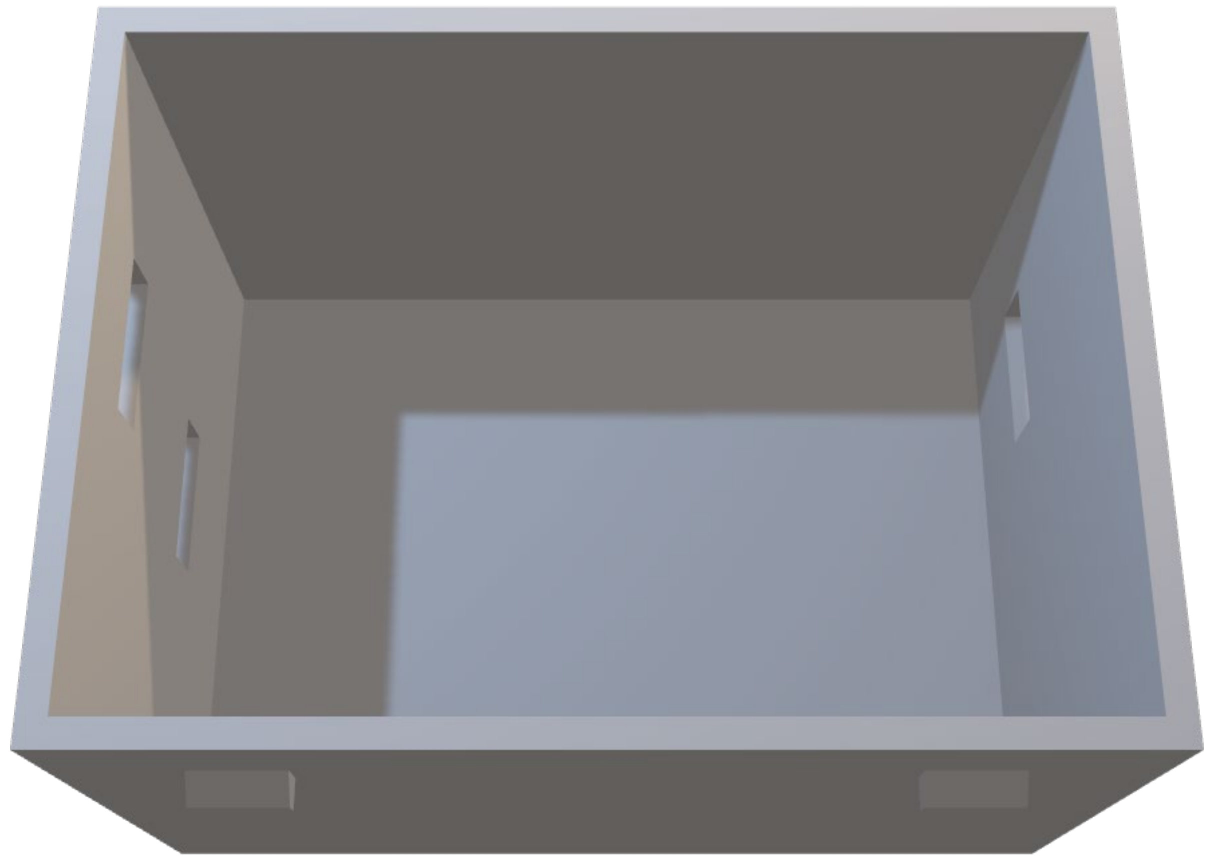
FDM printing works by heating a plastic filament to its melting temperature and then extruding the molten plastic through a fine nozzle.

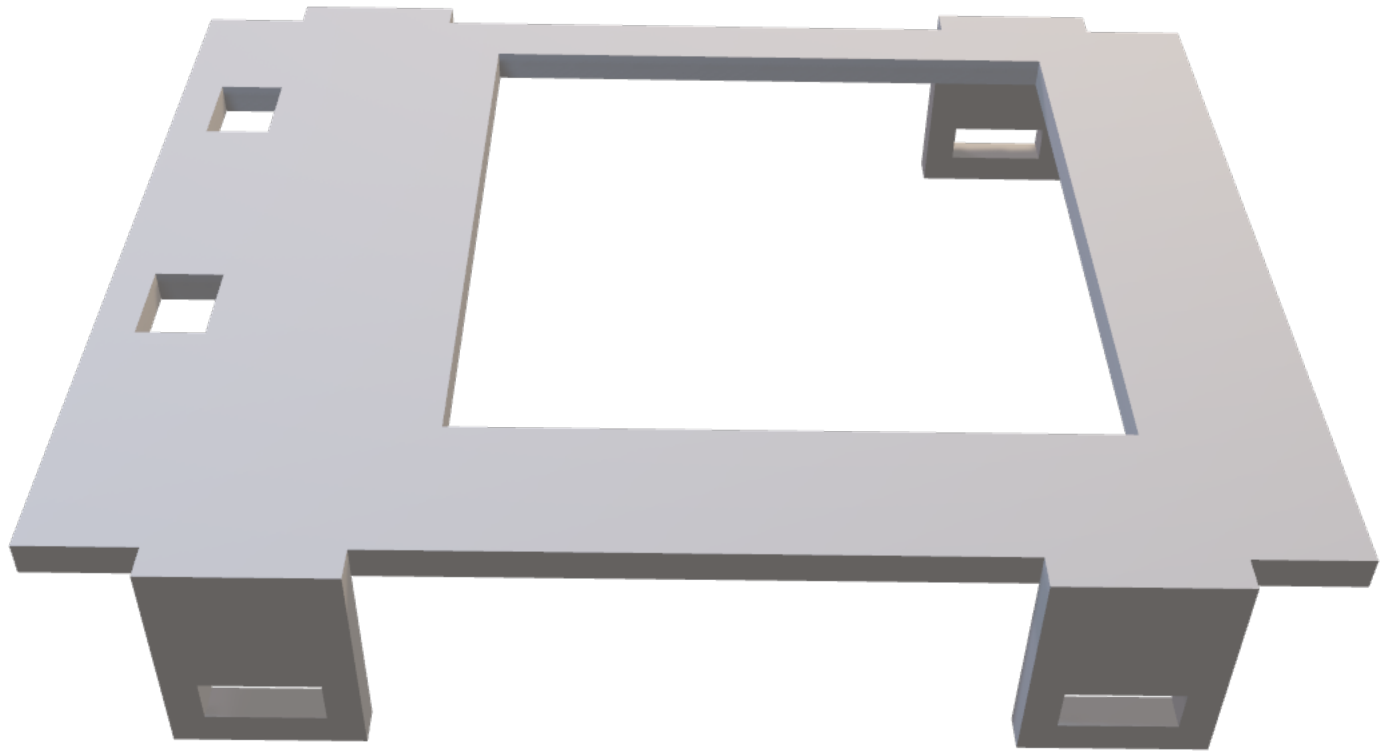
This process repeats, layer by layer, until the part is fully created.

Sometimes support material must be used for hallow parts and can be easily removed after.

14.

Case Model





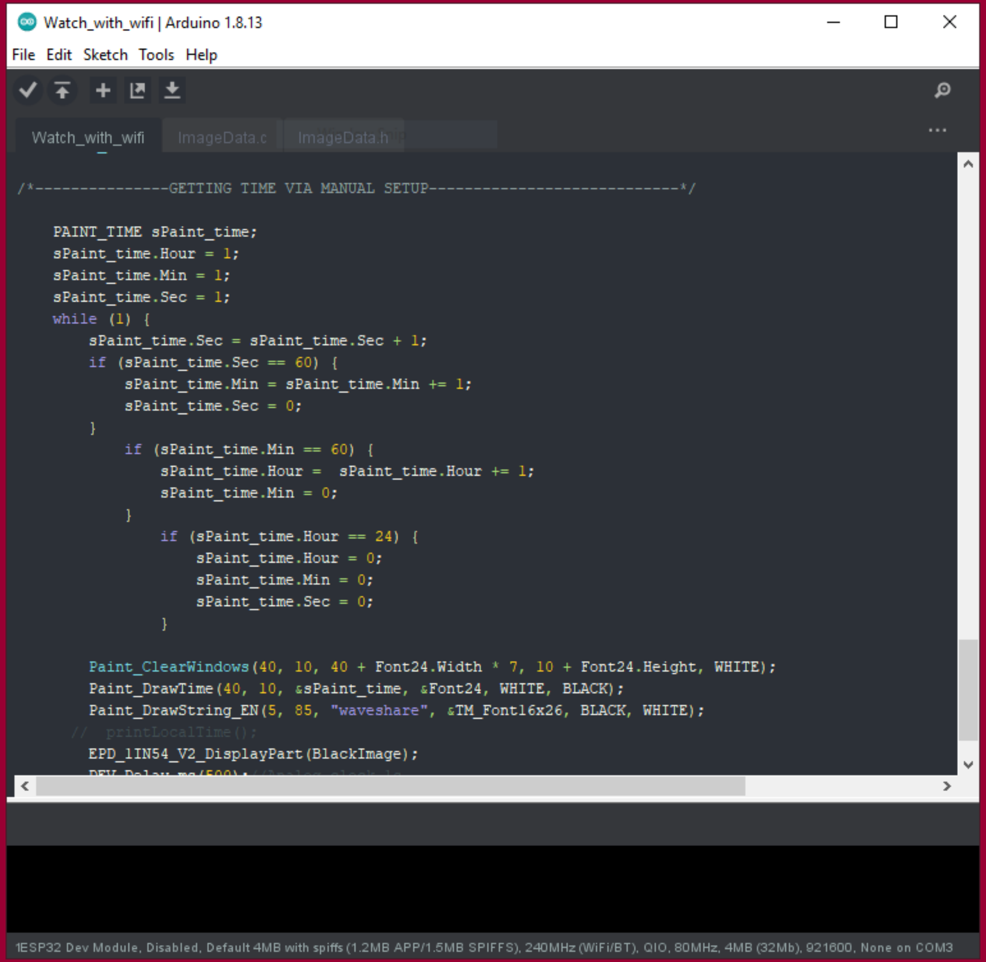
Time to Reflect!

3. What revisions would you make to the current case design?

4. What are the advantages to using 3D printing to create a case?

#SmartWatchKit

CODING



```
/*---------------GETTING TIME VIA MANUAL SETUP----------------------------*/

    PAINT_TIME sPaint_time;
    sPaint_time.Hour = 1;
    sPaint_time.Min = 1;
    sPaint_time.Sec = 1;
    while (1) {
        sPaint_time.Sec = sPaint_time.Sec + 1;
        if (sPaint_time.Sec == 60) {
            sPaint_time.Min = sPaint_time.Min += 1;
            sPaint_time.Sec = 0;
        }
            if (sPaint_time.Min == 60) {
                sPaint_time.Hour =  sPaint_time.Hour += 1;
                sPaint_time.Min = 0;
            }
                if (sPaint_time.Hour == 24) {
                    sPaint_time.Hour = 0;
                    sPaint_time.Min = 0;
                    sPaint_time.Sec = 0;
                }

        Paint_ClearWindows(40, 10, 40 + Font24.Width * 7, 10 + Font24.Height, WHITE);
        Paint_DrawTime(40, 10, &sPaint_time, &Font24, WHITE, BLACK);
        Paint_DrawString_EN(5, 85, "waveshare", &TM_Font16x26, BLACK, WHITE);
      //  printLocalTime();
        EPD_1IN54_V2_DisplayPart(BlackImage);
```

1ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3

Coding

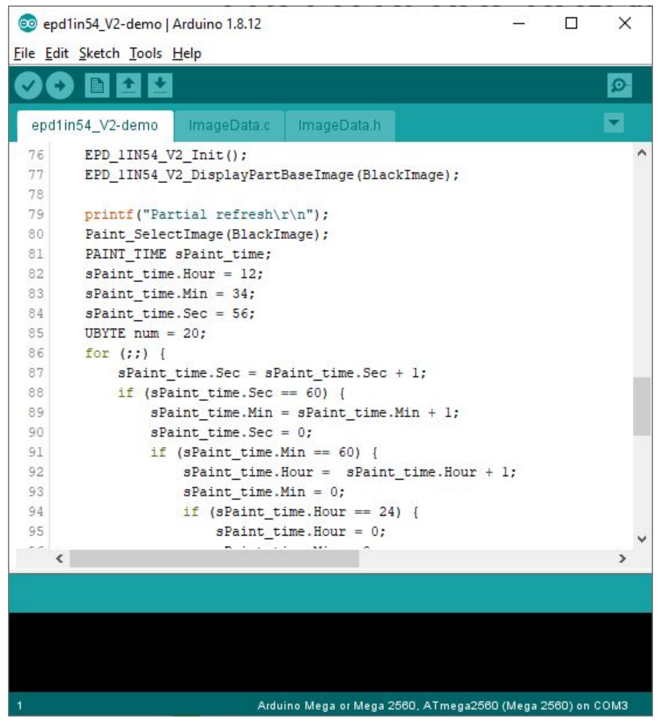
Let's you communicate with a computer using a programming language to give it instructions on what to do.

20.

These programming languages can be written with a type of software called IDE (Integrated Development Environment).

19.

There are hundreds of types of programming languages, but some of the most commonly used are C++, JAVA, C#, Python and Ruby.

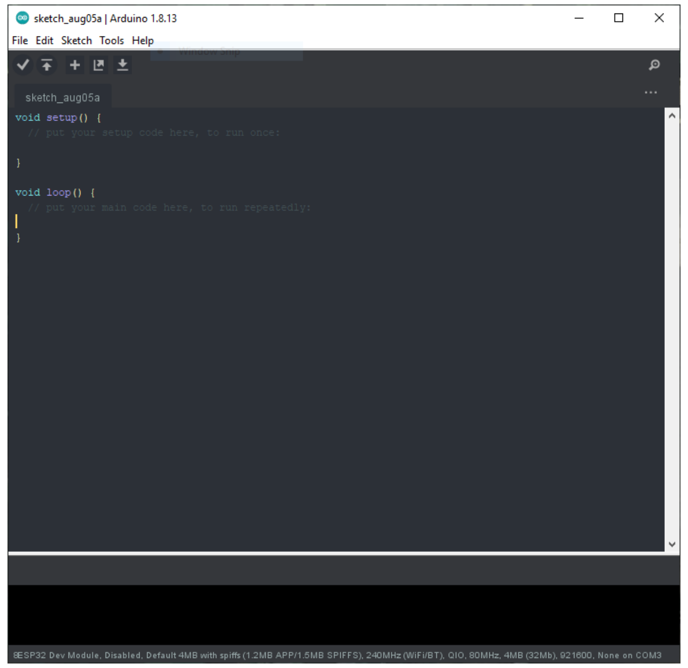

epd1in54_V2-demo | Arduino 1.8.12
File Edit Sketch Tools Help
epd1in54_V2-demo
ImageData.c
ImageData.h
Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM3

```
    EPD_1IN54_V2_Init();
    EPD_1IN54_V2_DisplayPartBaseImage(BlackImage);

    printf("Partial refresh\r\n");
    Paint_SelectImage(BlackImage);
    PAINT_TIME sPaint_time;
    sPaint_time.Hour = 12;
    sPaint_time.Min = 34;
    sPaint_time.Sec = 56;
    UBYTE num = 20;
    for (;;) {
        sPaint_time.Sec = sPaint_time.Sec + 1;
        if (sPaint_time.Sec == 60) {
            sPaint_time.Min = sPaint_time.Min + 1;
            sPaint_time.Sec = 0;
            if (sPaint_time.Min == 60) {
                sPaint_time.Hour =  sPaint_time.Hour + 1;
                sPaint_time.Min = 0;
                if (sPaint_time.Hour == 24) {
                    sPaint_time.Hour = 0;
```

Arduino

Arduino IDE

The Arduino IDE is an open-sourced software based in C/C++ that makes it easy to write code and upload it to the board. 21.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3

Libraries

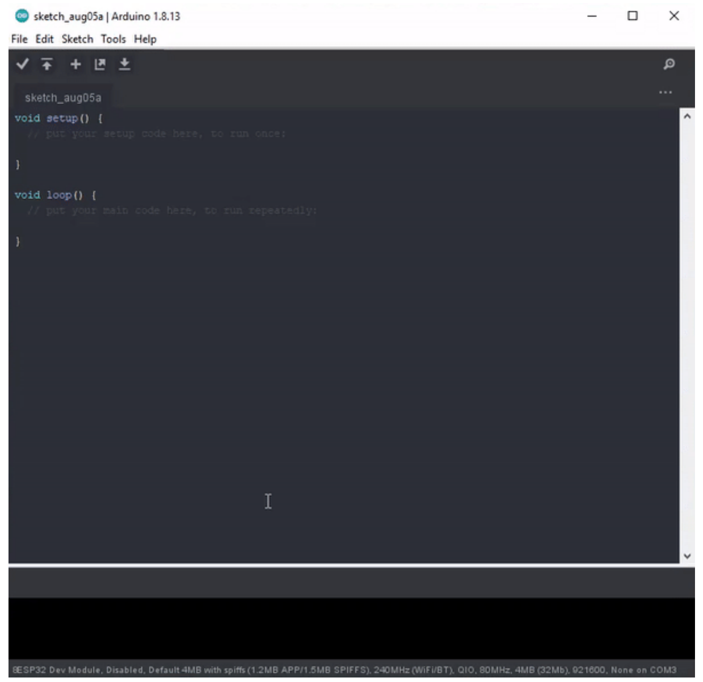
Libraries are files written in C or C++ which provide your sketches with extra functionality. This project will require the addition of different libraries. 24.

ARDUINO

22.

Sketches

A sketch is the name that Arduino uses for a program. It's the unit of code that is uploaded to and ran on the board. 23.



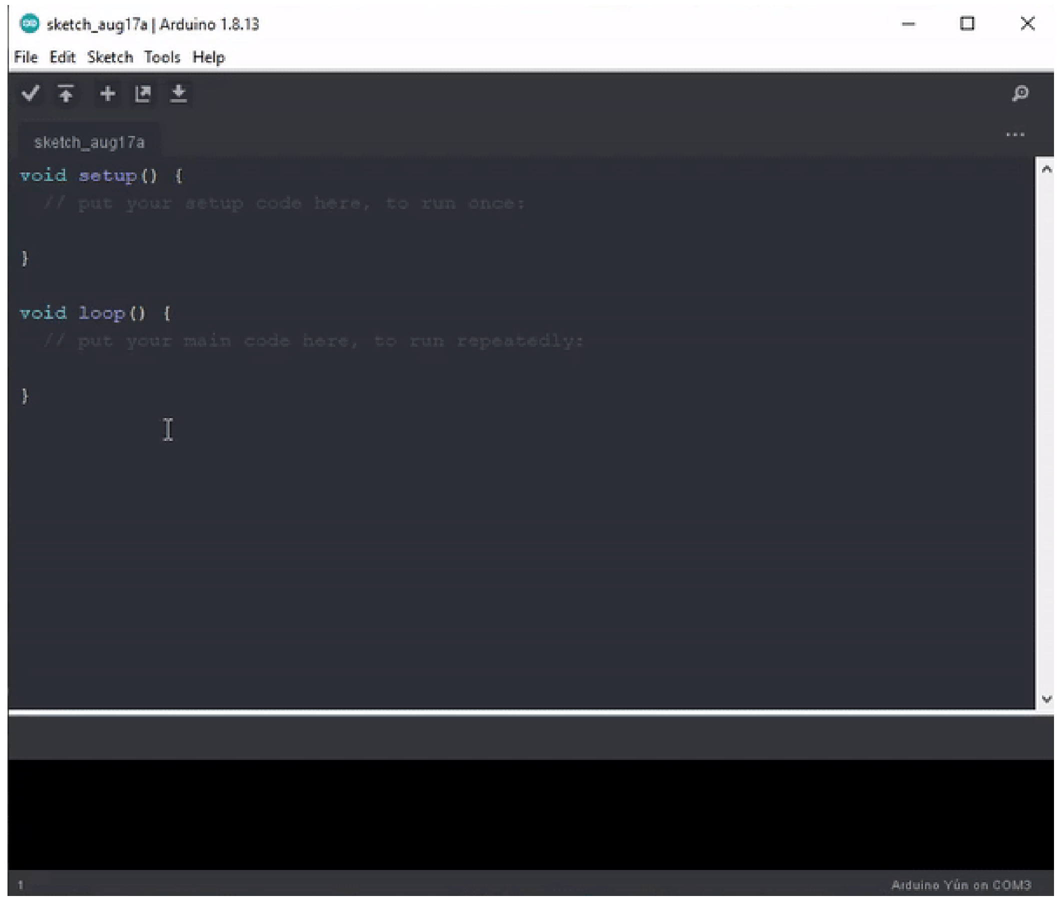
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3

Installing Board and Library Files

Installing and Setting up the Board

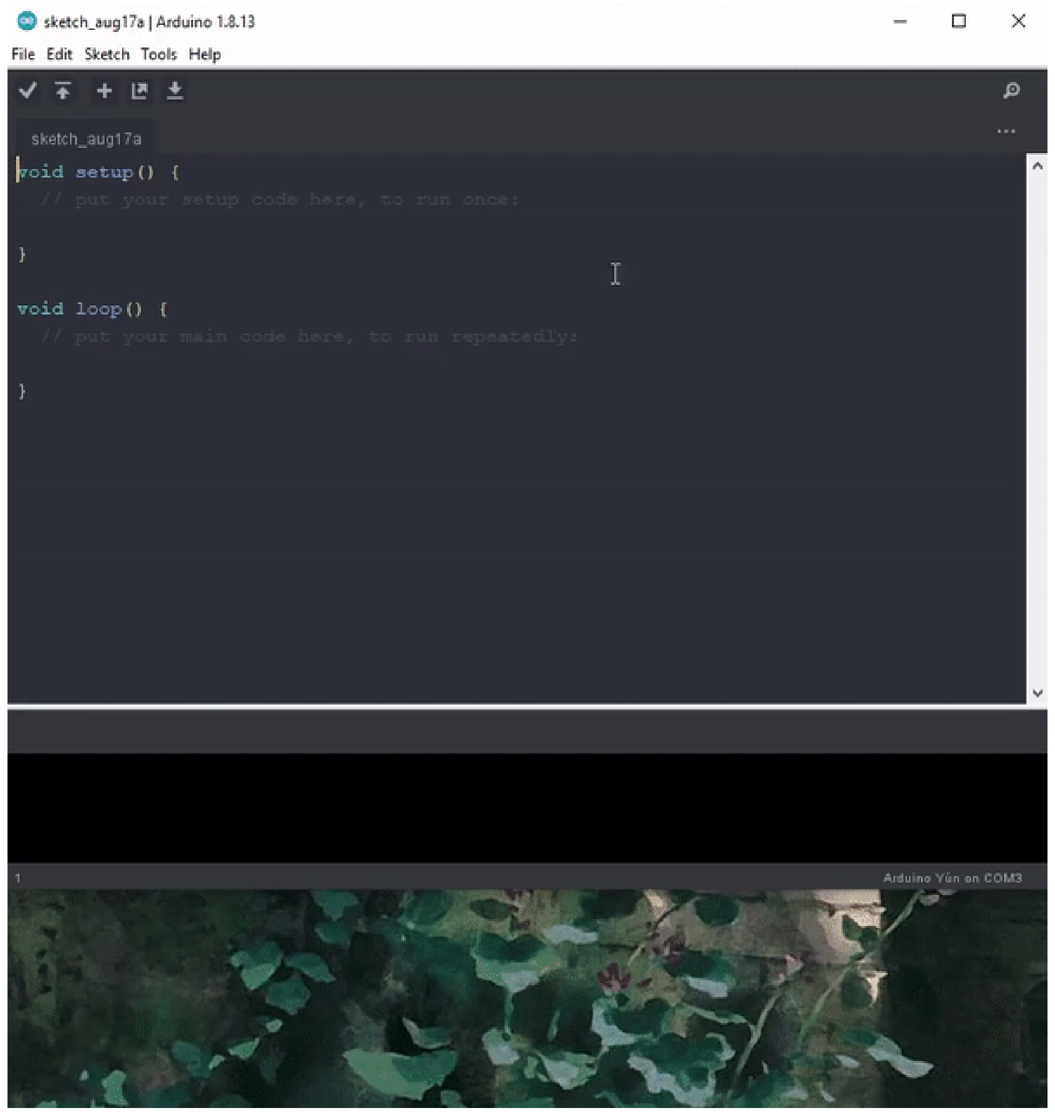
1. Open the Arduino IDE. Go to the Tools drop-down menu, then go down to Board, and then select Boards Manager.
2. In the Boards Manager window, search for ESP32, then select install.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Installing Board and Library Files

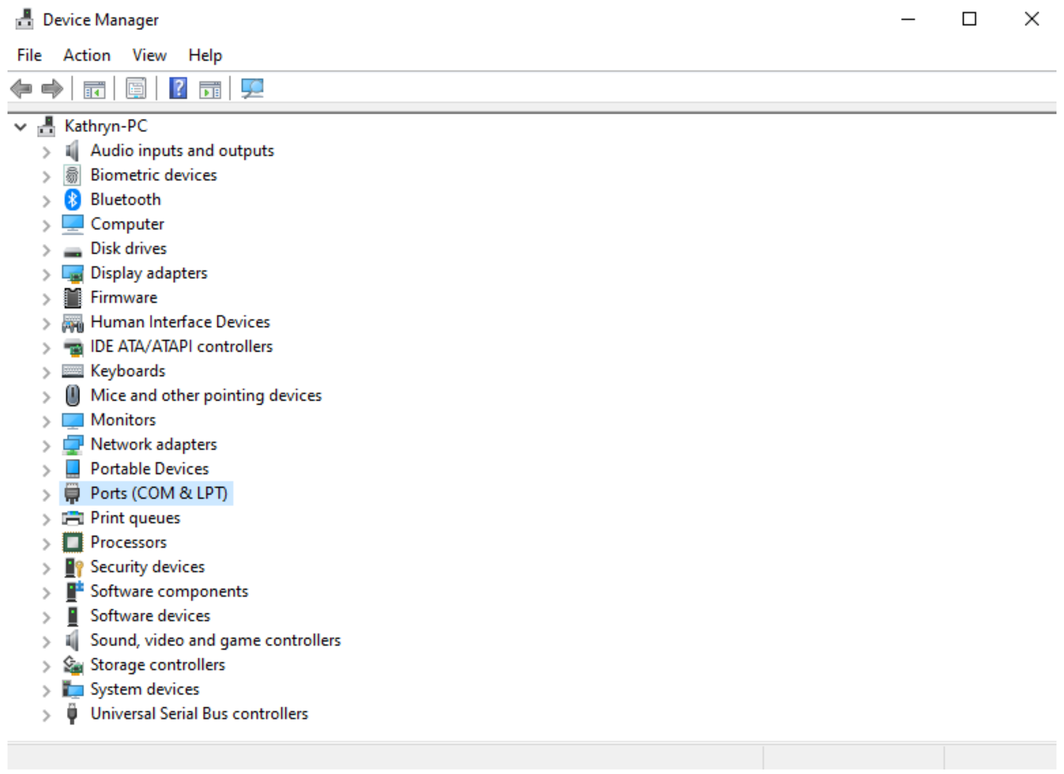
3. Go to the Tools drop-down menu, go down to Board, then to ESP32 Arduino and select the ESP32 Dev Module.

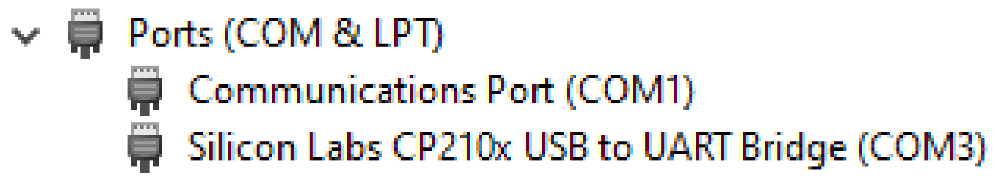


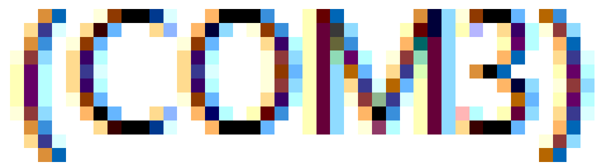
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Installing Board and Library Files

4. Plug in the board to your computer using the USB to micro USB cable.
5. Open the start menu on your PC and type in device manager, then hit enter.
6. In Device Manager go down Ports(COM & LPT) and hit the little arrow next to it to see all the serial ports. There you will see a port named Silicon Labs CP210X USB to UART Bridge. Look at the brackets to see what COM port number it is connected to.

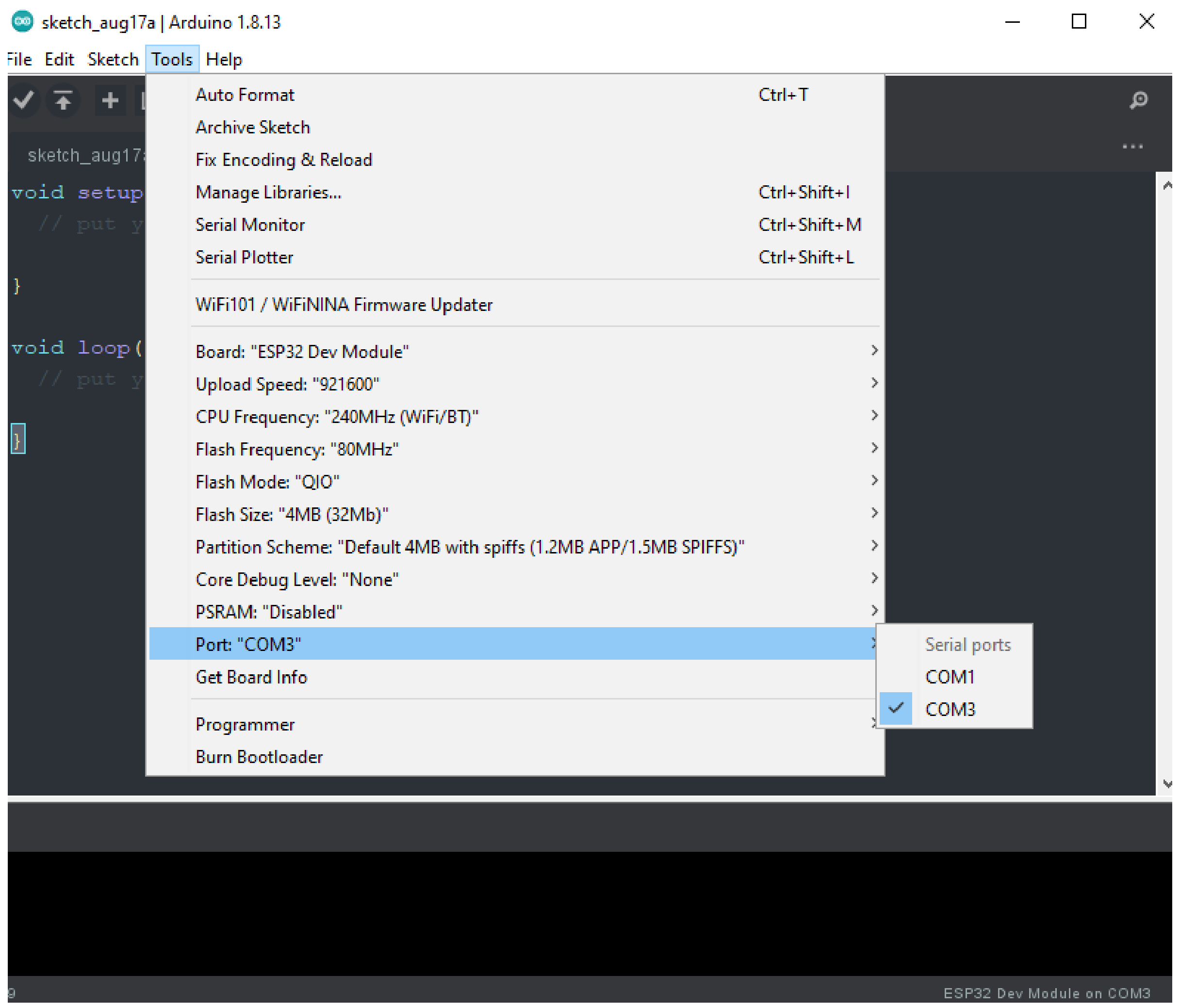



Ports (COM & LPT)
Communications Port (COM1)
Silicon Labs CP210x USB to UART Bridge (COM3)



Installing Board and Library Files

7. In the Arduino IDE, go to the Tools drop-down menu, go down to Ports, then select the appropriate serial port.



sketch_aug17a | Arduino 1.8.13

File Edit Sketch Tools Help

```
void setup
  // put y

}

void loop(
  // put y

}
```

Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater	
Board: "ESP32 Dev Module"	
Upload Speed: "921600"	
CPU Frequency: "240MHz (WiFi/BT)"	
Flash Frequency: "80MHz"	
Flash Mode: "QIO"	
Flash Size: "4MB (32Mb)"	
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	
Core Debug Level: "None"	
PSRAM: "Disabled"	
Port: "COM3"	
Get Board Info	
Programmer	
Burn Bootloader	

Serial ports
COM1
✓ COM3

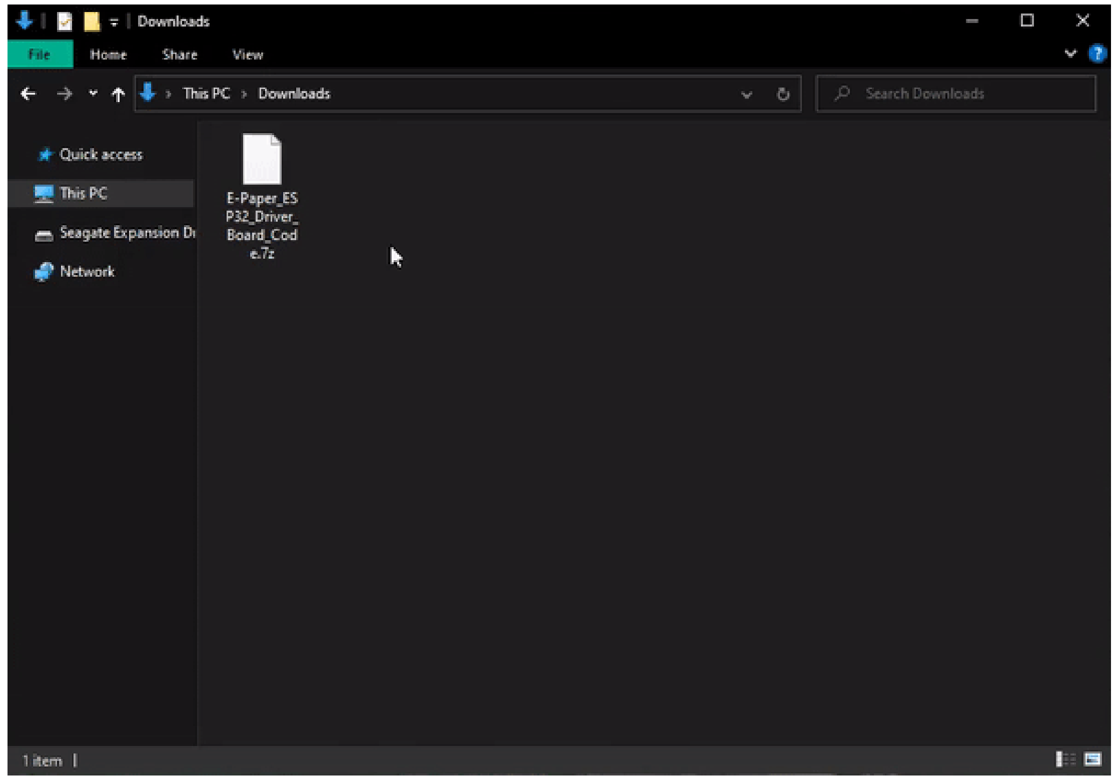
ESP32 Dev Module on COM3

The board is now ready to be used with the Arduino IDE!

Installing Board and Library Files

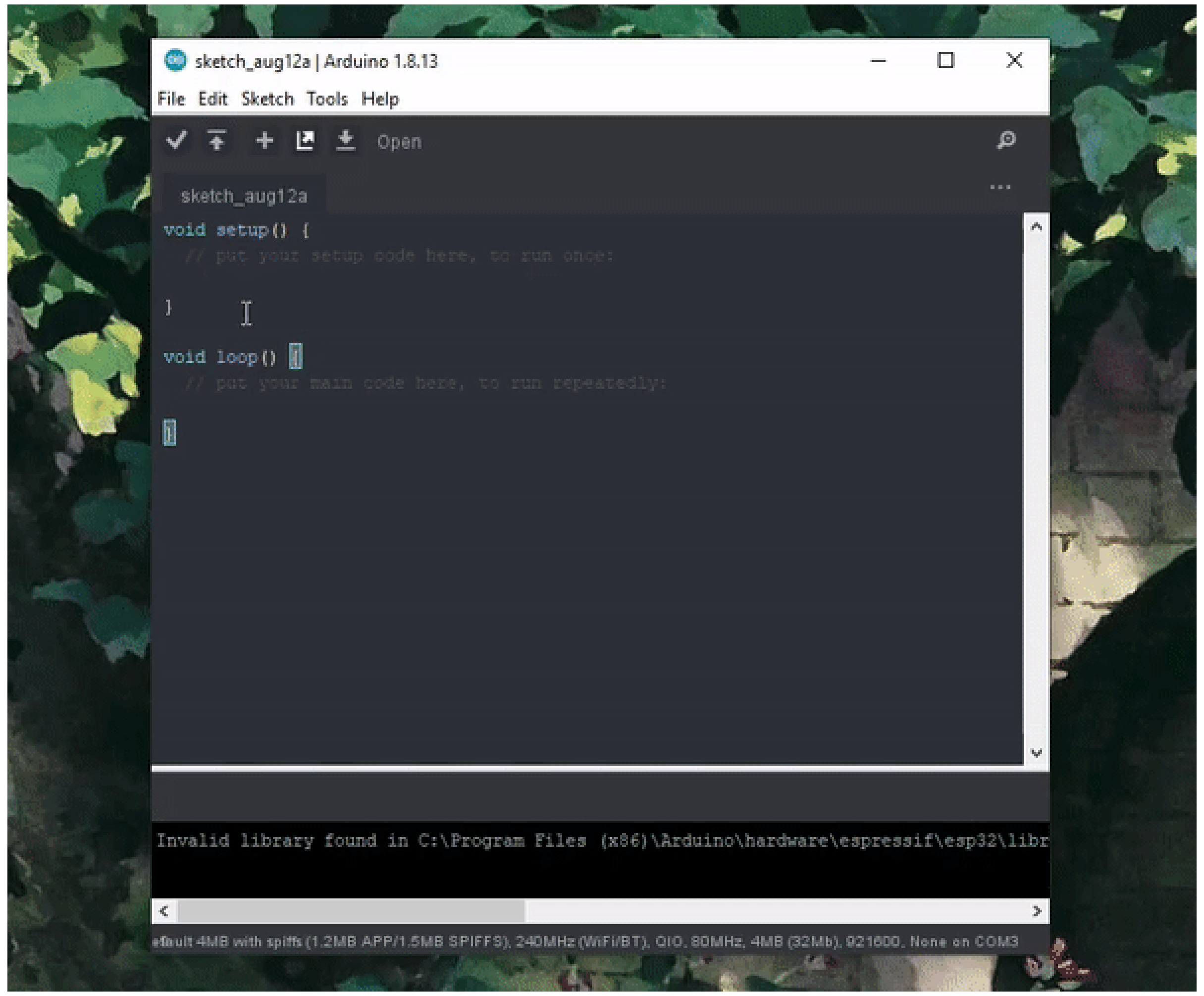
Installing Libraries

1. Download the E-Paper ESP32 Driver Board Code.
2. Extract the files from E-Paper_ESP32_Driver_Board_Code.
3. Open the E-Paper_ESP32_Driver_Board_Code extracted folder, then open the examples folder.
4. Make a .zip file of esp32-waveshare-epd folder.



Installing Board and Library Files

5. Open the Arduino IDE, go to the Sketch drop down menu, then go down to Include Library, from there select Add .ZIP Library.
6. In the file explorer, go to the file location where you created the .zip of the esp32-waveshare-epd folder and select it.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Invalid library found in C:\Program Files (x86)\Arduino\hardware\espressif\esp32\libr
```

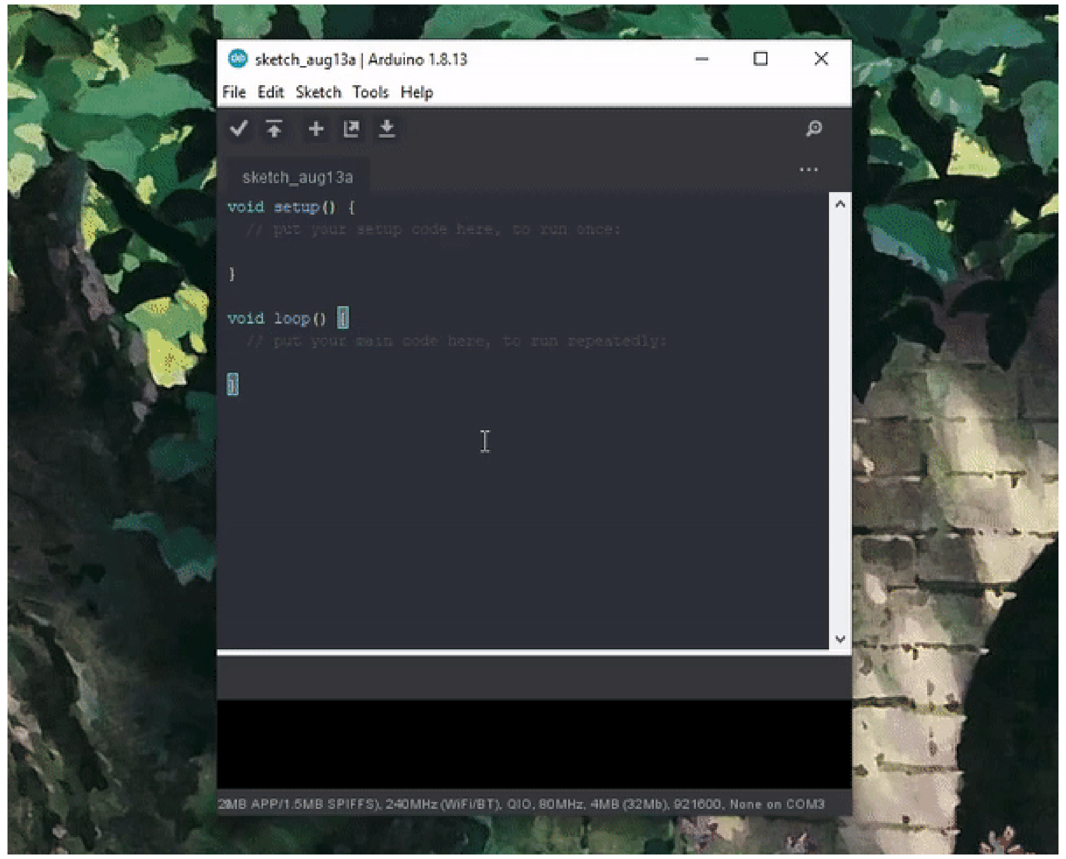
efault 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3

Installing Board and Library Files

7. Download the minigrafx, Time, ESP32 e-Paper Weather Display, Adafruit GFX, GxEPD2 and ArduinoJson libraries.

8. Open the Arduino IDE, go to the Sketch drop down menu, then go down to Include Library, from there select Add .ZIP Library.

9. In the file explorer, go to your downloads and select one of the .zip file of the libraries you just downloaded. Do this for each of the 6 libraries.



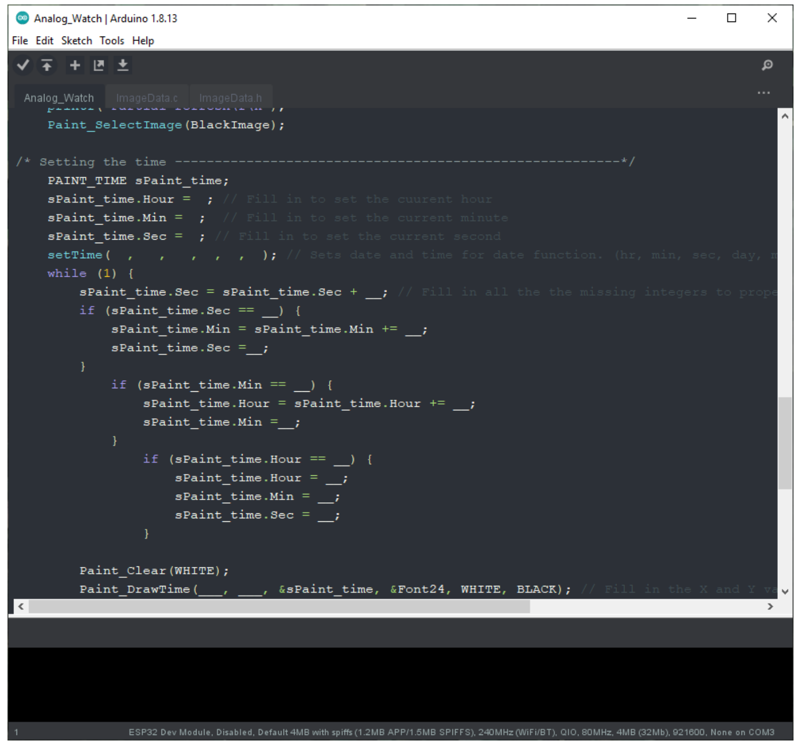
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3
```

The libraries are now installed, and you are ready to start coding!

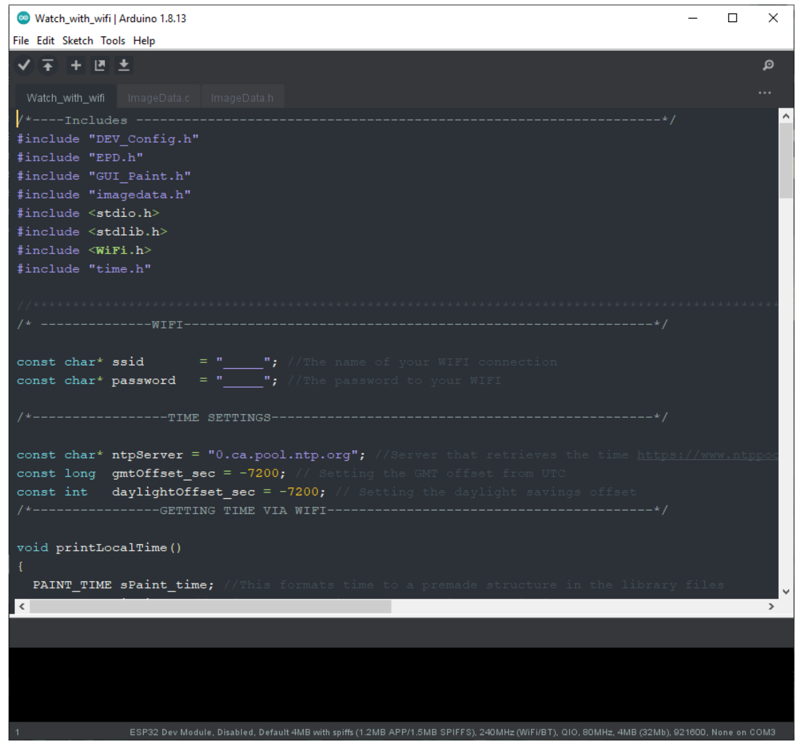
Example Code

This project will include 3 example codes, Analog Watch, Watch with WIFI, and Weather. Analog Watch and Watch with WIFI will contain activities within them to complete the code, while Weather is a showcase of some of the more intricate projects you can create with this hardware and libraries. For more example codes, and to see more of the board's capabilities go to the File drop-down menu in the Arduino IDE, then down to Examples, from there you can see example codes for ESP32 Dev Module and the libraries you added.



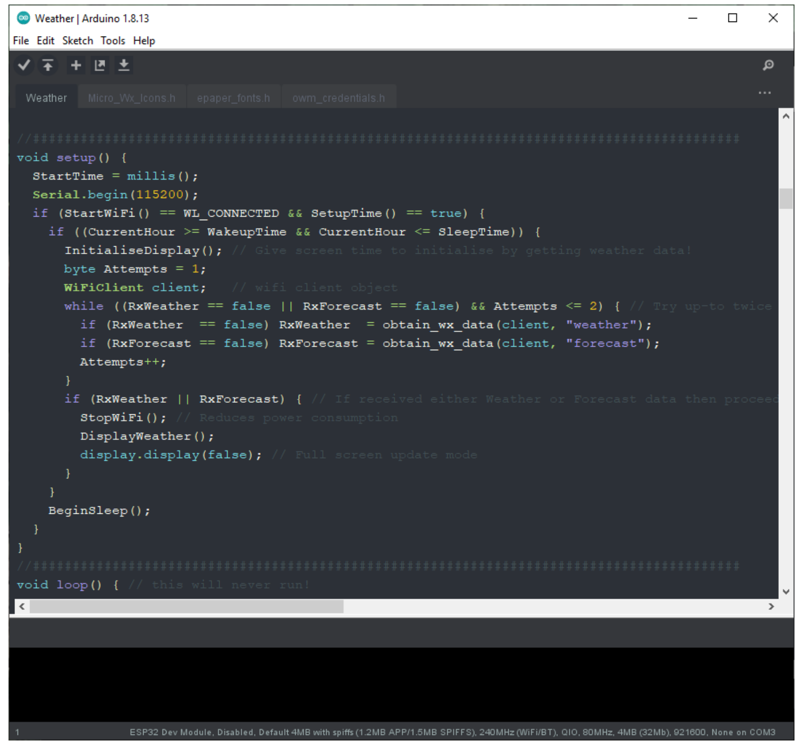
```
    Paint_SelectImage(BlackImage);

/* Setting the time ---------------------------------------------------------*/
    PAINT_TIME sPaint_time;
    sPaint_time.Hour =  ; // Fill in to set the cuurent hour
    sPaint_time.Min =  ;  // Fill in to set the current minute
    sPaint_time.Sec =  ; // Fill in to set the current second
    setTime(  ,   ,   ,  ,  ,  ); // Sets date and time for date function. (hr, min, sec, day, m
    while (1) {
        sPaint_time.Sec = sPaint_time.Sec + __; // Fill in all the the missing integers to prope
        if (sPaint_time.Sec == __) {
            sPaint_time.Min = sPaint_time.Min += __;
            sPaint_time.Sec =__;
        }
            if (sPaint_time.Min == __) {
                sPaint_time.Hour = sPaint_time.Hour += __;
                sPaint_time.Min =__;
            }
                if (sPaint_time.Hour == __) {
                    sPaint_time.Hour = __;
                    sPaint_time.Min = __;
                    sPaint_time.Sec = __;
                }

        Paint_Clear(WHITE);
        Paint_DrawTime(___, ___, &sPaint_time, &Font24, WHITE, BLACK); // Fill in the X and Y va
```

```
1        ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3
```



```
/*----Includes --------------------------------------------------------------*/
#include "DEV_Config.h"
#include "EPD.h"
#include "GUI_Paint.h"
#include "imagedata.h"
#include <stdio.h>
#include <stdlib.h>
#include <WiFi.h>
#include "time.h"

//*************************************************************************************
/* -------------WIFI---------------------------------------------------------*/

const char* ssid       = "_____"; //The name of your WIFI connection
const char* password   = "_____"; //The password to your WIFI

/*----------------TIME SETTINGS----------------------------------------------*/

const char* ntpServer = "0.ca.pool.ntp.org"; //Server that retrieves the time https://www.ntppoo
const long  gmtOffset_sec = -7200; // Setting the GMT offset from UTC
const int   daylightOffset_sec = -7200; // Setting the daylight savings offset
/*---------------GETTING TIME VIA WIFI---------------------------------------*/

void printLocalTime()
{
  PAINT_TIME sPaint_time; //This formats time to a premade structure in the library files
```

```
1        ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3
```



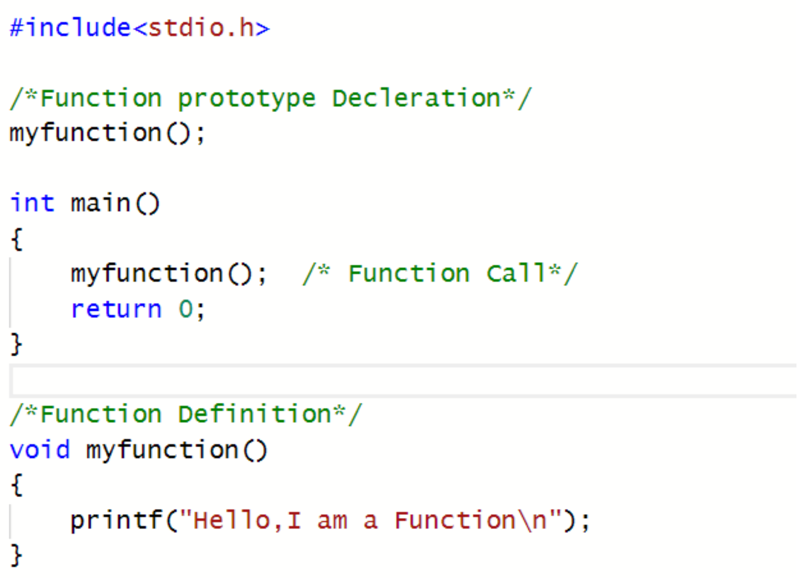
```
//#################################################################################
void setup() {
  StartTime = millis();
  Serial.begin(115200);
  if (StartWiFi() == WL_CONNECTED && SetupTime() == true) {
    if ((CurrentHour >= WakeupTime && CurrentHour <= SleepTime)) {
      InitialiseDisplay(); // Give screen time to initialise by getting weather data!
      byte Attempts = 1;
      WiFiClient client;   // wifi client object
      while ((RxWeather == false || RxForecast == false) && Attempts <= 2) { // Try up-to twice
        if (RxWeather  == false) RxWeather  = obtain_wx_data(client, "weather");
        if (RxForecast == false) RxForecast = obtain_wx_data(client, "forecast");
        Attempts++;
      }
      if (RxWeather || RxForecast) { // If received either Weather or Forecast data then proceed
        StopWiFi(); // Reduces power consumption
        DisplayWeather();
        display.display(false); // Full screen update mode
      }
    }
    BeginSleep();
  }
}
//#################################################################################
void loop() { // this will never run!
```

```
1        ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM3
```

Code Functions

Function

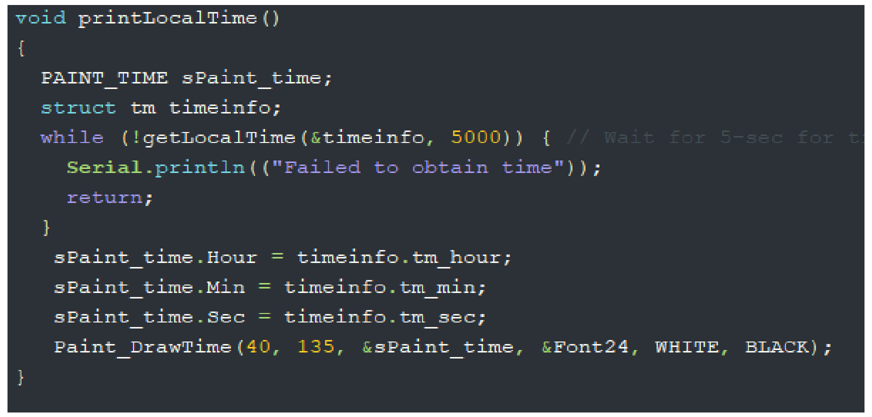
A function is a set of statements that take inputs, do some specific computation and produces output. 25.



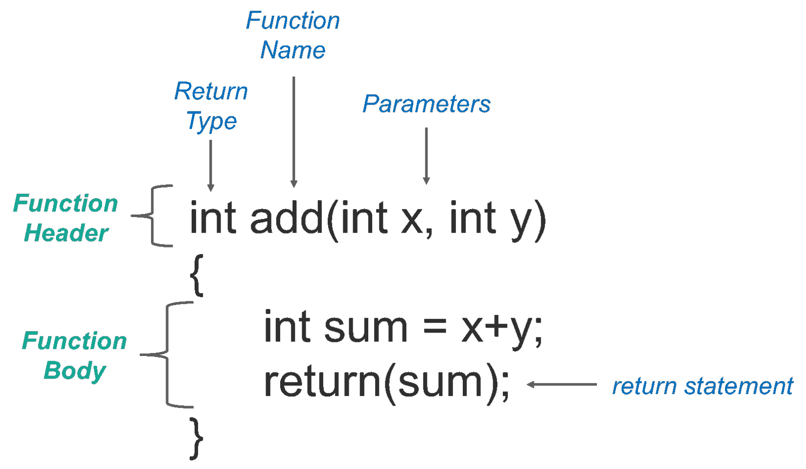
```
#include<stdio.h>

/*Function prototype Decleration*/
myfunction();

int main()
{
    myfunction();  /* Function Call*/
    return 0;
}

/*Function Definition*/
void myfunction()
{
    printf("Hello,I am a Function\n");
}
```

26.

A function definition consists of a **return type**, **function name**, **parameters**, and **function body**. 27.



```
void printLocalTime()
{
  PAINT_TIME sPaint_time;
  struct tm timeinfo;
  while (!getLocalTime(&timeinfo, 5000)) { // Wait for 5-sec for t
    Serial.println(("Failed to obtain time"));
    return;
  }
   sPaint_time.Hour = timeinfo.tm_hour;
   sPaint_time.Min = timeinfo.tm_min;
   sPaint_time.Sec = timeinfo.tm_sec;
   Paint_DrawTime(40, 135, &sPaint_time, &Font24, WHITE, BLACK);
}
```



```
int add(int x, int y)
{
      int sum = x+y;
      return(sum);
}
```

Functions can be used to divide up your code into individual tasks. This makes it easier to troubleshoot your code as well as make it easier to read.

28.

Code Functions

A **return type** is the data type of the value the function returns. Ex. **int, float**, **char** etc. In cases where the desired function operation does not return a value, the keyword **void** is used.

A **Function name** is the actual name of the function. The function name will be used to call the function in the main routine.

```
return_type function_name( parameter list ) {
   body of the function
}
```

29.

The **Function body** contains a collection of statements that define what the function does.

Parameters are used as placeholders for the data going into a function. When a function is used, you pass a value to the parameter. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional, a function may contain no parameters. 27.

Code Functions

Library Functions

Library functions are built-in functions in the libraries that only need to be called upon to be used. Here’s a list of the functions in the libraries used in our example code, and how a they work:

Initialization and screen clearing

DEV_Module_Init(); - This tells the board what pins are being used by the e-paper display. It also sets the data rate in bit per second for serial data transmission.

EPD_1IN54_V2_Init(); - This initializes the e-paper register, which holds values of internal operations, such as the address of the instruction being executed, and the data being processed.

EPD_1IN54_V2_Clear(); - This clears any data left on the e-paper display.

EPD_1IN54_Display(ImageName); - This sends image buffer in the boards RAM to the e-paper display.

DEV_Delay_ms(milliseconds); - This is used to suspend execution of a program for a particular time.

Code Functions

Paint_NewImage(ImageName, Width, Height, Rotation, Colour); - This takes an image in the cache and displays it on the screen. Width and height can be set between 1-200 pixels. Rotation can be set as 0, 90, 180, and 270 degrees. Colour can be set as BLACK or WHITE depending on background colour you would like.

```
Paint_NewImage(BlackImage, EPD_1IN54_V2_WIDTH, EPD_1IN54_V2_HEIGHT, 180, WHITE);
```

Paint_SelectImage(ImageName); - This allows you to select an image from the cache.

Paint_Clear(Colour); - This clears the display setting it to all white or all black, using WHITE or BLACK in the brackets.

Paint_ClearWindows(X Start, Y Start, X End, Y End, Colour); - This clears a specific part of the display. X and Y start, are the starting coordinates. X and Y end are the end coordinates. Everything between those point will clear to the set colour (WHITE or BLACK).

```
Paint_ClearWindows(15, 65, 15 + Font20.Width * 7, 65 + Font20.Height, WHITE);
```

Code Functions

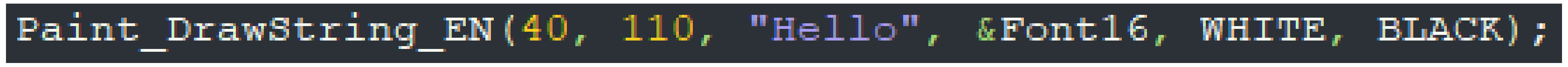
Displaying Strings

Parameters for using the print strings functions are
(X, Y, ”__ ”, &Font, Font colour, Background colour);

- X and Y are the coordinates of where on the display you want the number/string printed. For both X and Y, the number can be between 1-200, as the screen is 200 x 200 pixels.
- In the quotation marks goes the number/string you want to print. If you wanted to print a variable, you would need to do (”%d”, variableName) for int variable and (”%s”, variableName) for a string variable, instead of the quotation marks.
- &Font can be set as &Font8, &Font12, &Font16, &Font20, or &Font24 depending on the size of font you would like.
- Font colour can be set as WHITE or BLACK, the same goes for background colour.

Paint_DrawString_EN(X, Y, “__”, &Font, Font colour, Background colour); - This function will print a string.

Paint_DrawNum(X, Y, “__”, &Font, Font colour, Background colour); - This function will print a number.



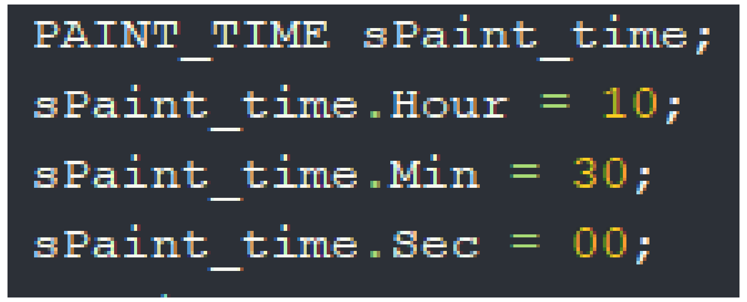
```
Paint_DrawString_EN(40, 110, "Hello", &Font16, WHITE, BLACK);
```

Code Functions

Displaying Time

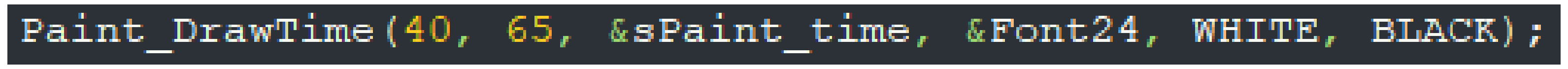
If you want to print the time on the display you will first need to declare PAINT_TIME sPaint_time;. PAINT_TIME is a structure. A structure is a collection of variables (can be of different types) under a single name. In this case PAINT_TIME is used to define time attributes, i.e. hours, minutes, and seconds. sPaint_time is a preset variable to used to access the structure. 30.

To access the variables in the structure you will need to use sPaint_time.Hour, sPaint_time.Min, and sPaint_time.Sec.



```
PAINT_TIME sPaint_time;
sPaint_time.Hour = 10;
sPaint_time.Min = 30;
sPaint_time.Sec = 00;
```

Paint_DrawTime(X, Y, &sPaint_time, &Font, Font colour, Background colour); - This function will print the time on the display.



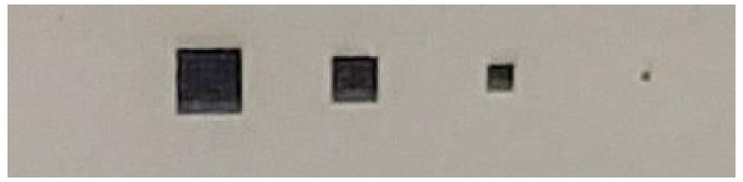
```
Paint_DrawTime(40, 65, &sPaint_time, &Font24, WHITE, BLACK);
```

Code Functions

Displaying Shapes

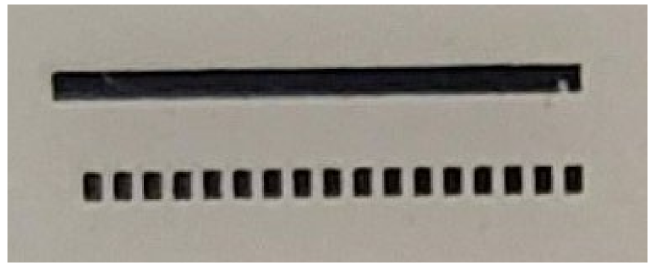
Paint_DrawPoint(X, Y, Colour, Point size, DOT_STYLE_DFT); - This function will print a point. Point size can be adjusted by using DOT_PIXEL_1X1, DOT_PIXEL_2X2, DOT_PIXEL_3X3, etc. all the way up to 8X8.

```
Paint_DrawPoint(5, 10, BLACK, DOT_PIXEL_1X1, DOT_STYLE_DFT);
```



Paint_DrawLine(X Start, Y Start, X End, Y End, Colour, Point size, Line style); - This function will print a line from the starting coordinates to the end coordinates. There are 2 different line styles for this function. The first being LINE_STYLE_SOLID, which will print a solid line. The second being LINE_STYLE_DOTTED which will print a dotted line.

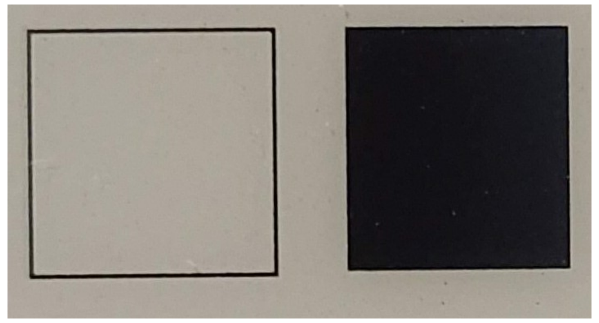
```
Paint_DrawLine(20, 10, 70, 60, BLACK, DOT_PIXEL_1X1, LINE_STYLE_SOLID);
```



Code Functions

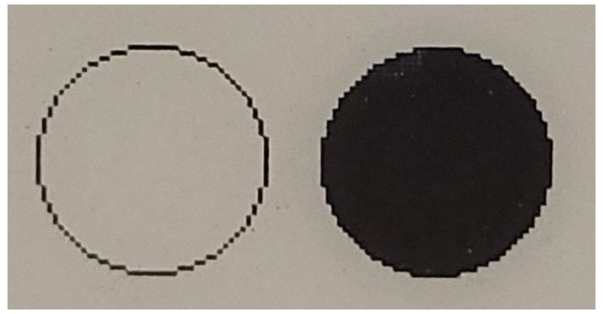
Paint_DrawRectangle(X Start, Y Start, X End, Y End, Colour, Point size, Fill style); - This function will print a rectangle from the starting coordinates on the top left, to the end coordinates on the bottom right There are 2 different fill styles. The first being DRAW_FILL_EMPTY, which will leave the rectangle empty. The second being DRAW_FILL_FULL which will completely fill in the rectangle black.

```
Paint_DrawRectangle(85, 10, 130, 60, BLACK, DOT_PIXEL_1X1, DRAW_FILL_FULL);
```



Paint_DrawLine(X , Y , Radius, Colour, Point size, Fill style); - This function will print a circle, with X and Y being the centre point.

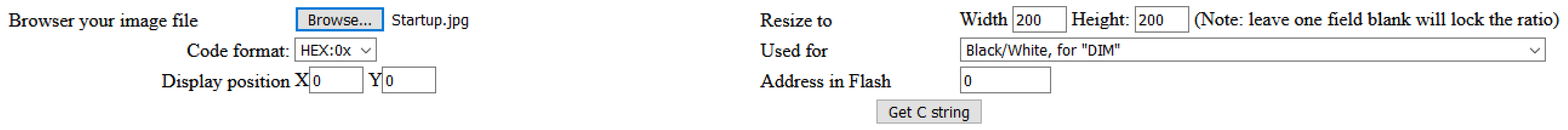
```
Paint_DrawCircle(170, 35, 20, BLACK, DOT_PIXEL_1X1, DRAW_FILL_EMPTY);
```



Startup Image

Follow these steps if you like to change the startup image that displays when the watch boots up.

1. Find an image. Try pick a simple image with little colour variation as it will lose details once it is converted to black and white. The image must also be upside down so that when it is uploaded to the watch it is displayed in the correct orientation. Also make sure the image is one these formats - .gif/.jpg/.jpeg/.png.
2. Convert the image in a hex code array. This website will do the conversion for you. Make sure you have the following settings before you click Get C string. Code format is set to HEX:0x, Resize to is set to Width 200, Height 200, and that Used for is set to Black/White. For "DIM".



Browser your image file [Browse...] Startup.jpg
Code format: HEX:0x
Display position X 0 Y 0
Resize to Width 200 Height: 200 (Note: leave one field blank will lock the ratio)
Used for Black/White, for "DIM"
Address in Flash 0

Startup Image

3. Copy all the hex code from the Image data box.

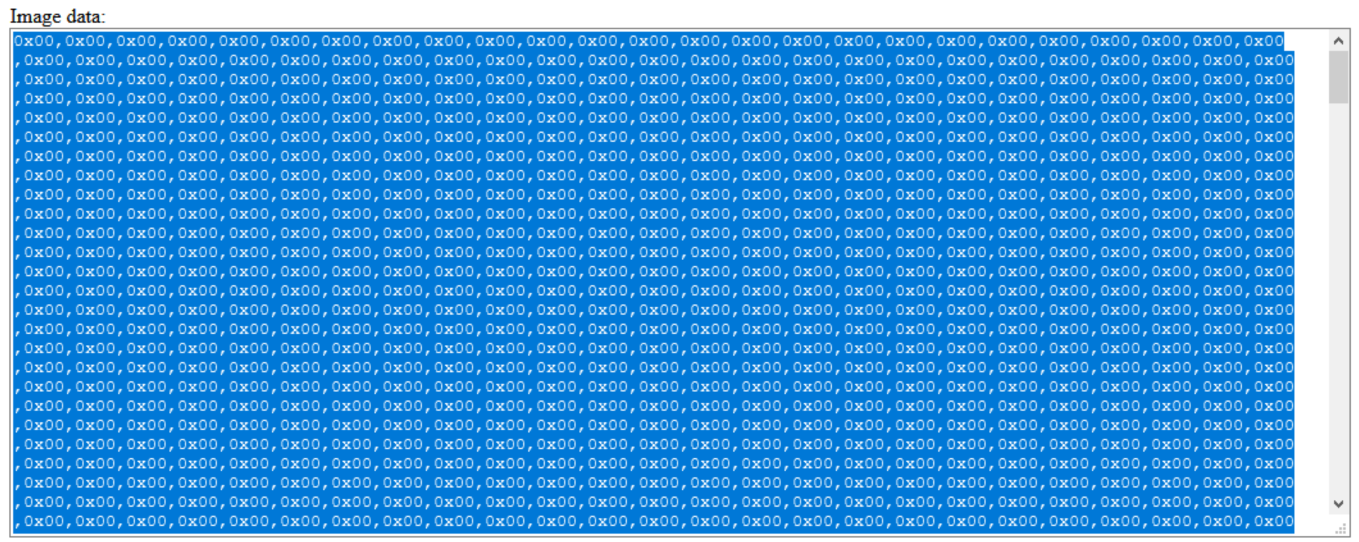
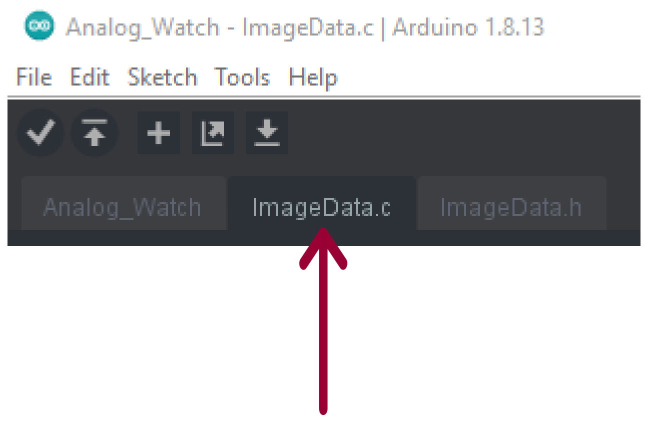


Image data:

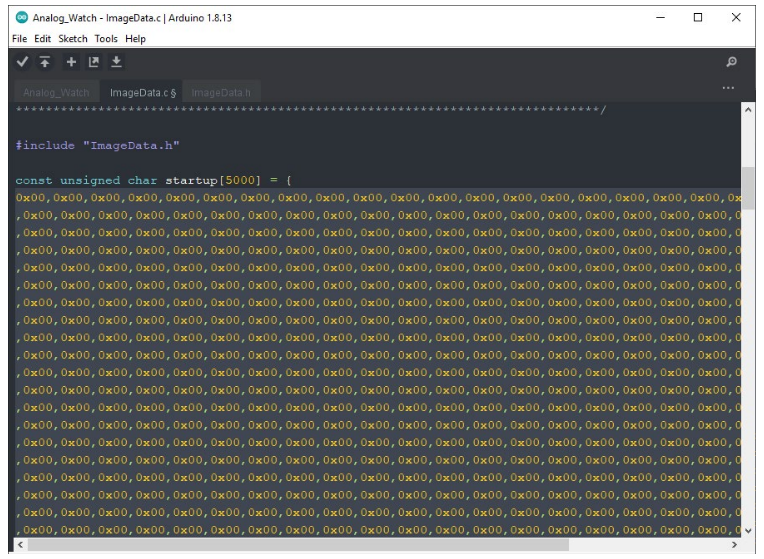
```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
```

4. Open the either the Analog Watch code or Watch with WIFI (which ever one you like to modify). Once it is open, go to the ImageData.c tab.


Analog_Watch - ImageData.c | Arduino 1.8.13
File Edit Sketch Tools Help
Analog_Watch
ImageData.c
ImageData.h

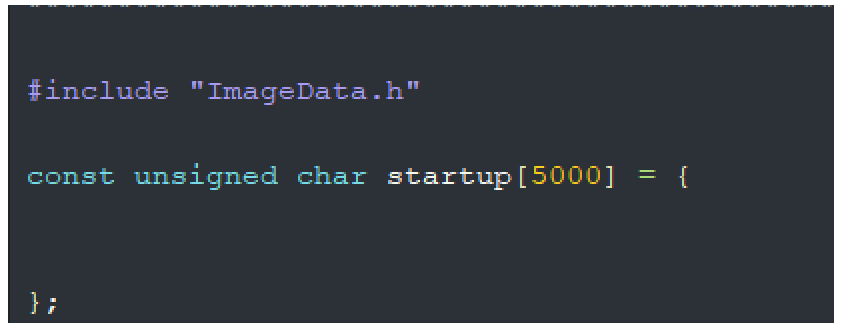
Startup Image

5. Highlight and delete all the hex code between const unsigned char startup[5000] = { and };.



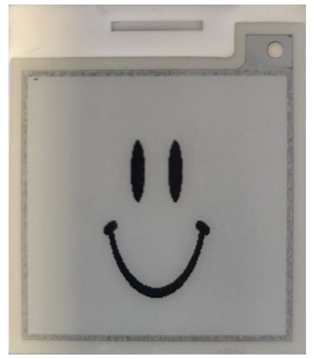
```
*******************************************************************************/

#include "ImageData.h"

const unsigned char startup[5000] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
```

Analog_Watch - ImageData.c | Arduino 1.8.13
File Edit Sketch Tools Help
Analog_Watch
ImageData.c §
ImageData.h



```
#include "ImageData.h"

const unsigned char startup[5000] = {


};
```

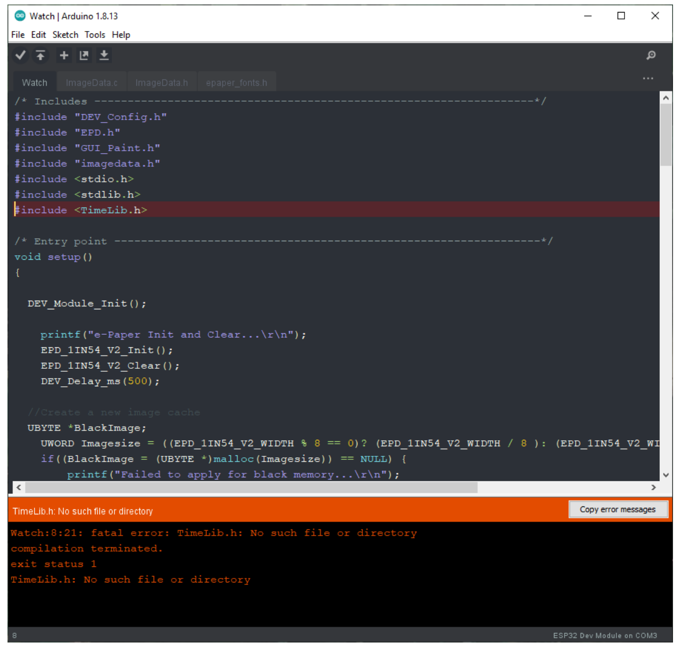
6. Take the hex code you copied from the image data box on the website and paste it between the curly brackets. Upload the code to the board and you should see your new startup image!



Troubleshooting

Here are solutions to commonly ran into issues while compiling the code:

1. **No such file or directory Error**. This is due to a library being missing. When the error appears, it will highlight the missing libraries .h file in the includes section. Once you know what library is missing, simply go to include library under the sketch menu, and add that libraries .zip file.

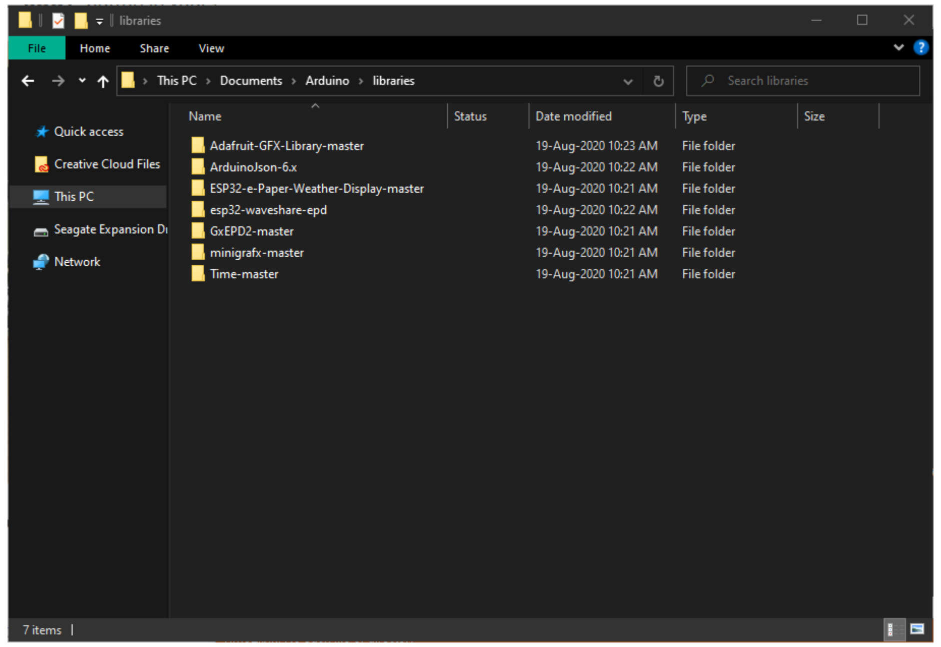


```
/* Includes ------------------------------------------------------------------*/
#include "DEV_Config.h"
#include "EPD.h"
#include "GUI_Paint.h"
#include "imagedata.h"
#include <stdio.h>
#include <stdlib.h>
#include <TimeLib.h>

/* Entry point ----------------------------------------------------------------*/
void setup()
{

  DEV_Module_Init();

    printf("e-Paper Init and Clear...\r\n");
    EPD_1IN54_V2_Init();
    EPD_1IN54_V2_Clear();
    DEV_Delay_ms(500);

  //Create a new image cache
  UBYTE *BlackImage;
    UWORD Imagesize = ((EPD_1IN54_V2_WIDTH % 8 == 0)? (EPD_1IN54_V2_WIDTH / 8 ): (EPD_1IN54_V2_WI
    if((BlackImage = (UBYTE *)malloc(Imagesize)) == NULL) {
        printf("Failed to apply for black memory...\r\n");
```

TimeLib.h: No such file or directory

```
Watch:8:21: fatal error: TimeLib.h: No such file or directory
compilation terminated.
exit status 1
TimeLib.h: No such file or directory
```

8 ESP32 Dev Module on COM3



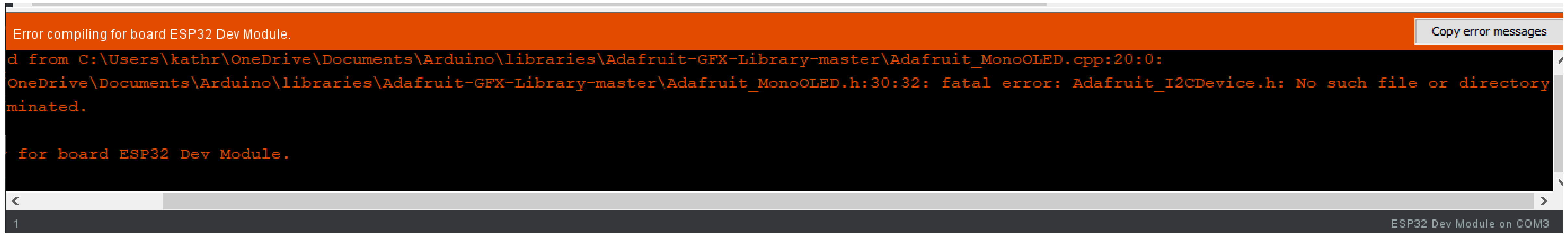
Name	Status	Date modified	Type	Size
Adafruit-GFX-Library-master		19-Aug-2020 10:23 AM	File folder	
ArduinoJson-6.x		19-Aug-2020 10:22 AM	File folder	
ESP32-e-Paper-Weather-Display-master		19-Aug-2020 10:21 AM	File folder	
esp32-waveshare-epd		19-Aug-2020 10:22 AM	File folder	
GxEPD2-master		19-Aug-2020 10:21 AM	File folder	
minigrafx-master		19-Aug-2020 10:21 AM	File folder	
Time-master		19-Aug-2020 10:21 AM	File folder	

7 items

To make sure you have all the libraries installed go to your documents folder on you PC, then to Arduino, then to Libraries (C:\Users\user\Documents\Arduino\libraries), and make sure all 7 libraries are there from slides 31 and 32.

Troubleshooting

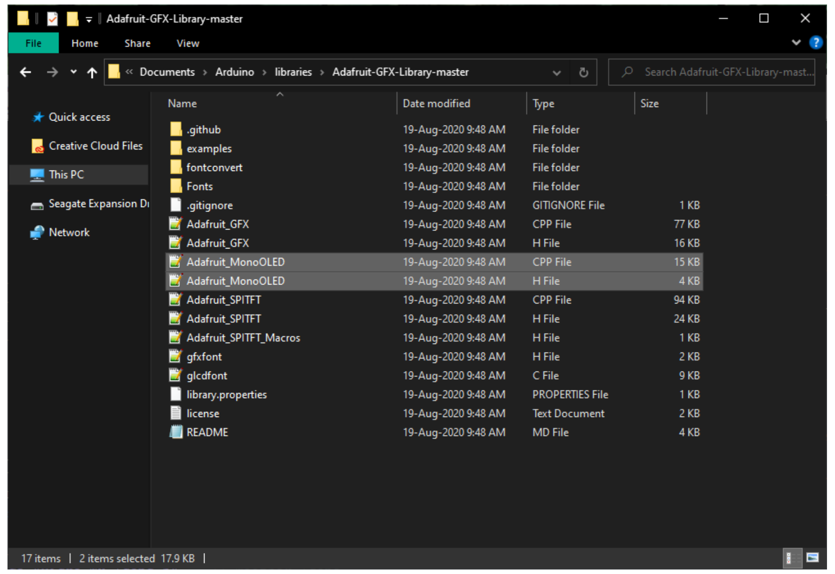
2. **Error compiling for board ESP32 DEV Module. Fatal error: Adafruit_12CDevice.h: no such file or directory.** This is due to the IDE trying to read 2 files in the Adafruit GFX library meant for OLED displays.



Error compiling for board ESP32 Dev Module.

```
d from C:\Users\kathr\OneDrive\Documents\Arduino\libraries\Adafruit-GFX-Library-master\Adafruit_MonoOLED.cpp:20:0:
OneDrive\Documents\Arduino\libraries\Adafruit-GFX-Library-master\Adafruit_MonoOLED.h:30:32: fatal error: Adafruit_I2CDevice.h: No such file or directory
minated.

 for board ESP32 Dev Module.
```

To fix this, go to your documents folder on you PC, then to Arduino, then to Libraries, then to Adafruit-GFX-Libraray Master (C:\Users\user\Documents\Arduino\libraries\Adafruit-GFX-Library-master). Once there, delete both Adafruit_MonoOLED files.



Adafruit-GFX-Library-master

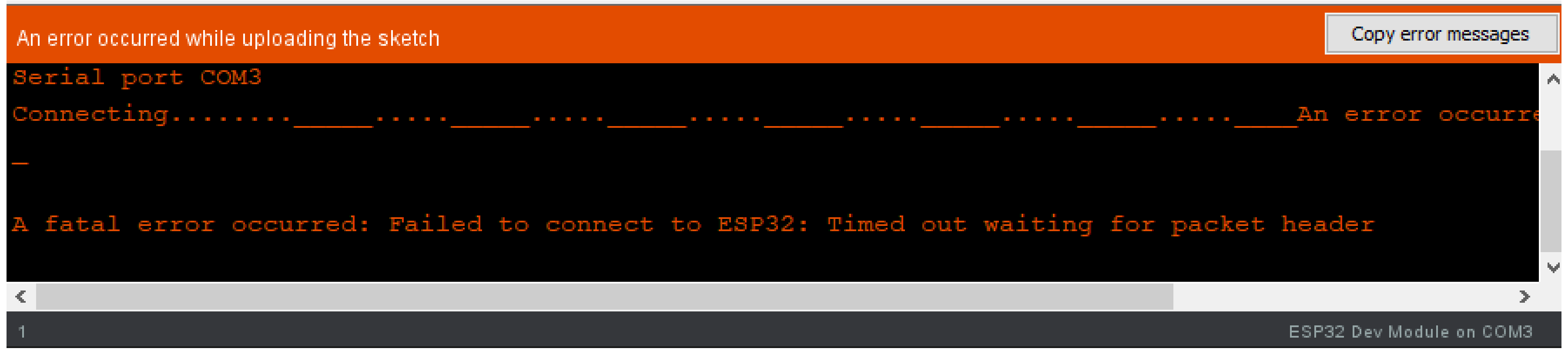
Documents › Arduino › libraries › Adafruit-GFX-Library-master

Name	Date modified	Type	Size
.github	19-Aug-2020 9:48 AM	File folder	
examples	19-Aug-2020 9:48 AM	File folder	
fontconvert	19-Aug-2020 9:48 AM	File folder	
Fonts	19-Aug-2020 9:48 AM	File folder	
.gitignore	19-Aug-2020 9:48 AM	GITIGNORE File	1 KB
Adafruit_GFX	19-Aug-2020 9:48 AM	CPP File	77 KB
Adafruit_GFX	19-Aug-2020 9:48 AM	H File	16 KB
Adafruit_MonoOLED	19-Aug-2020 9:48 AM	CPP File	15 KB
Adafruit_MonoOLED	19-Aug-2020 9:48 AM	H File	4 KB
Adafruit_SPITFT	19-Aug-2020 9:48 AM	CPP File	94 KB
Adafruit_SPITFT	19-Aug-2020 9:48 AM	H File	24 KB
Adafruit_SPITFT_Macros	19-Aug-2020 9:48 AM	H File	1 KB
gfxfont	19-Aug-2020 9:48 AM	H File	2 KB
glcdfont	19-Aug-2020 9:48 AM	C File	9 KB
library.properties	19-Aug-2020 9:48 AM	PROPERTIES File	1 KB
license	19-Aug-2020 9:48 AM	Text Document	2 KB
README	19-Aug-2020 9:48 AM	MD File	4 KB

17 items | 2 items selected 17.9 KB |

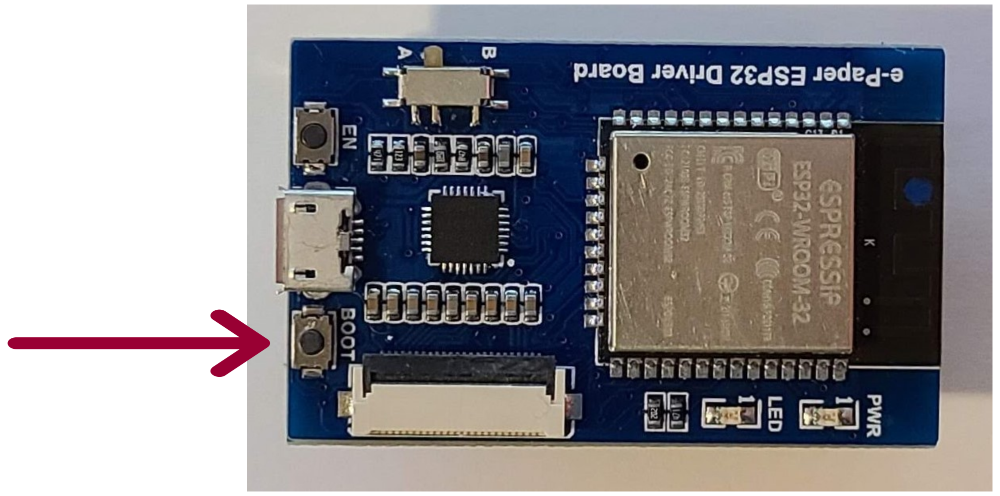
Troubleshooting

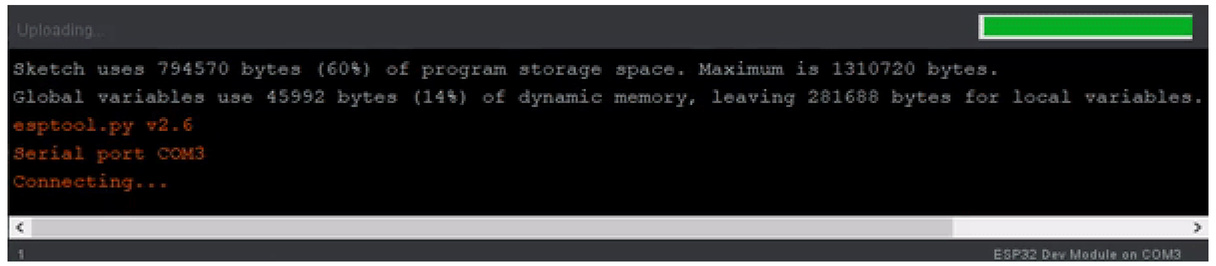
3. **A fatal error occurred: Failed to connect to ESP32: Timed out waiting for packet header.** This is due to the IDE not being able to connect to the board.



```
An error occurred while uploading the sketch                    Copy error messages
Serial port COM3
Connecting........_____....._____....._____....._____....._____....._____....._____An error occurre
_

A fatal error occurred: Failed to connect to ESP32: Timed out waiting for packet header
1                                                               ESP32 Dev Module on COM3
```

To get around this bug, when the code is compiled, and you see Connecting… in the console, hold the boot button on the driver board until you see the code is being written to the board on the console.





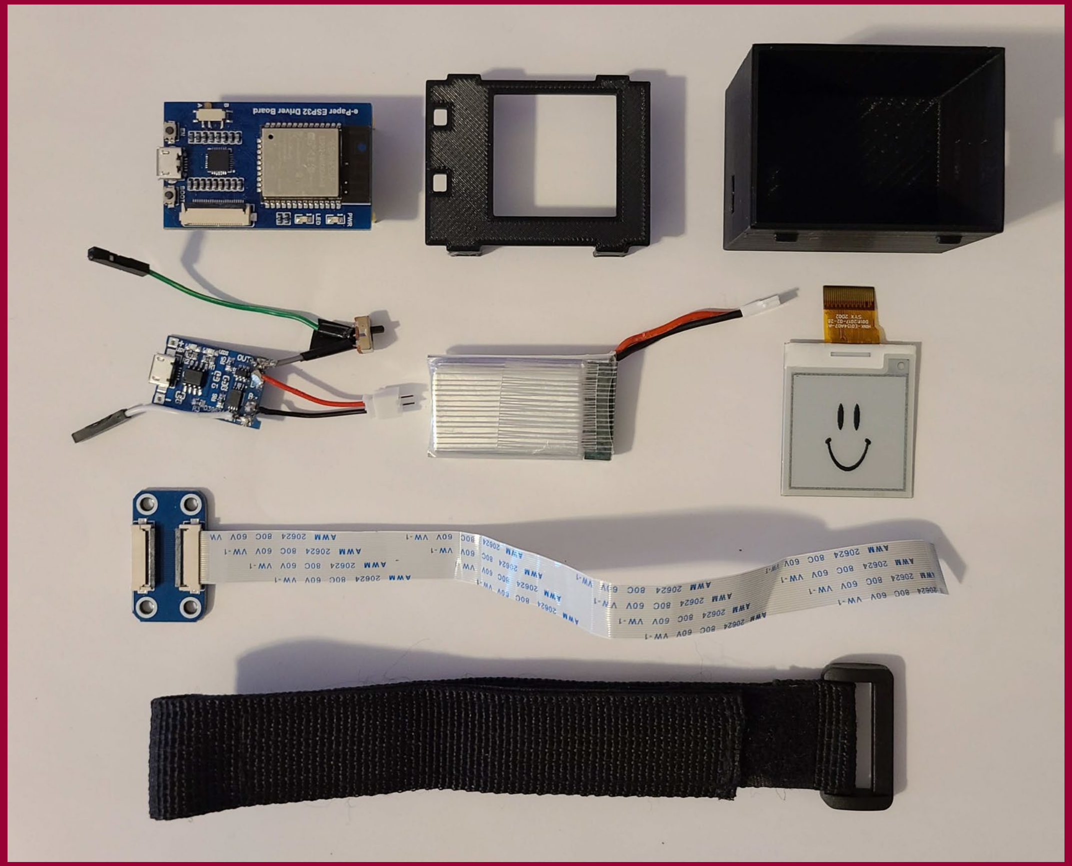
```
Uploading...
Sketch uses 794570 bytes (60%) of program storage space. Maximum is 1310720 bytes.
Global variables use 45992 bytes (14%) of dynamic memory, leaving 281688 bytes for local variables.
esptool.py v2.6
Serial port COM3
Connecting...
1                                                               ESP32 Dev Module on COM3
```

Time to Reflect!

5. What other functionality could you add to the watch?

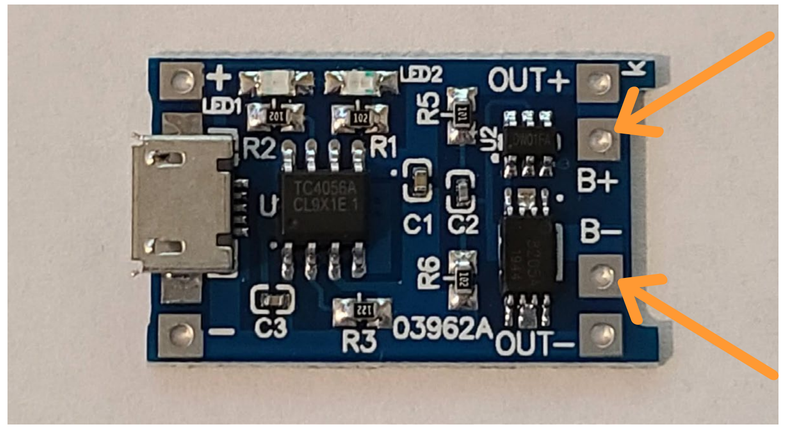
6. How could you integrate this code into other projects?

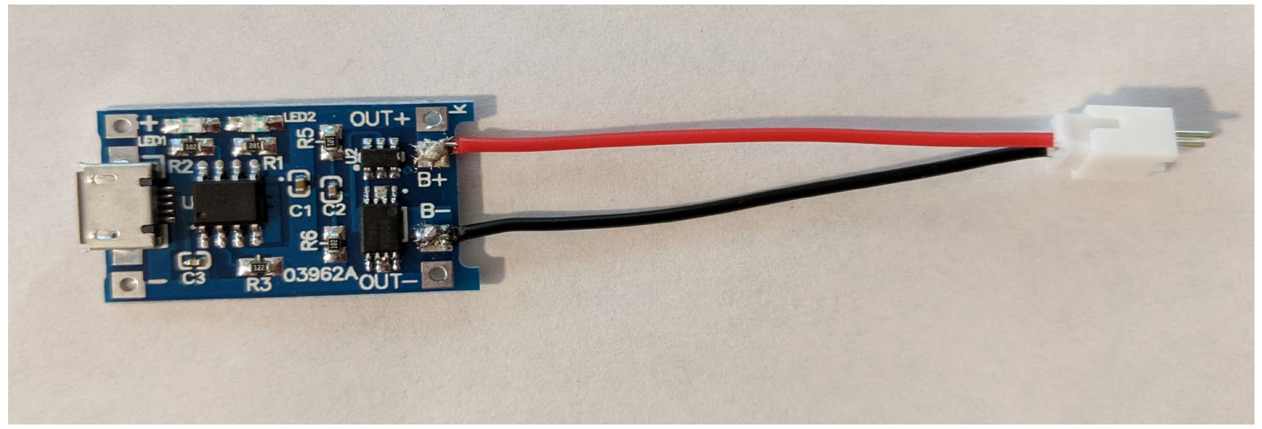
WATCH ASSEMBLY



Watch Assembly

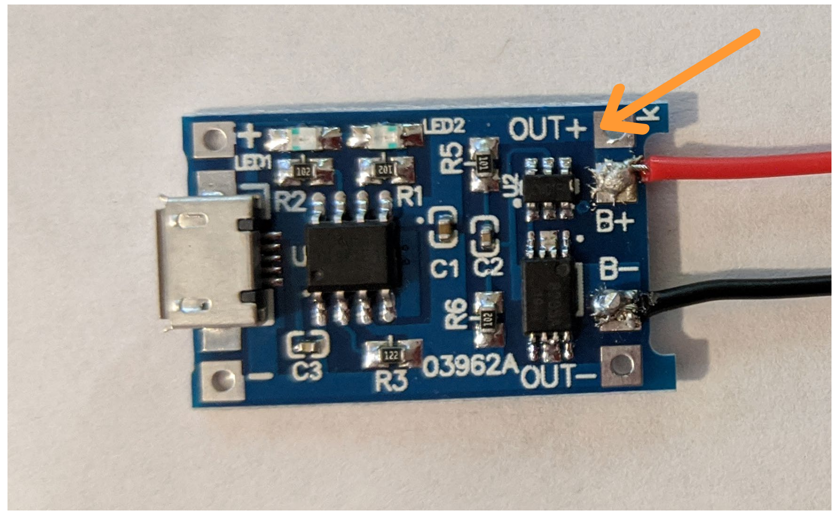
Wiring the TP4056 Charging Board

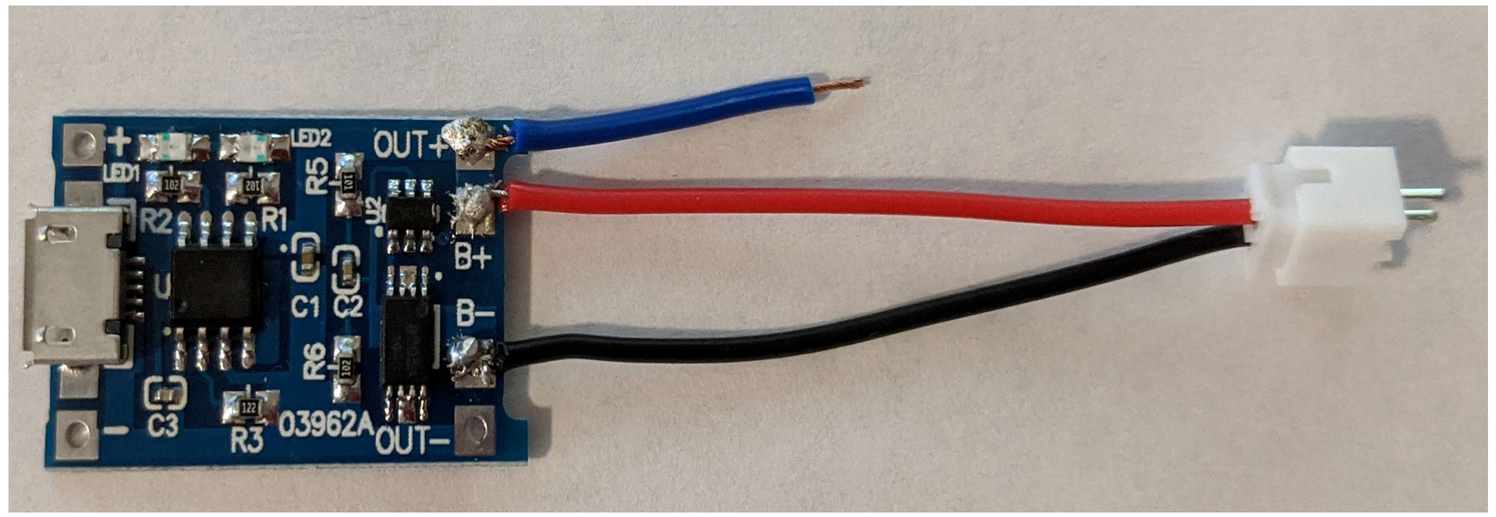




To begin wiring the TP4056 board we will start by soldering the Mini Micro Jst 2.0 Ph 2-Pin Connector Male Plug to the battery plated-through holes on the board. To match the polarity on the battery, make sure to solder the red wire to the positive hole, and the black wire to the negative hole.

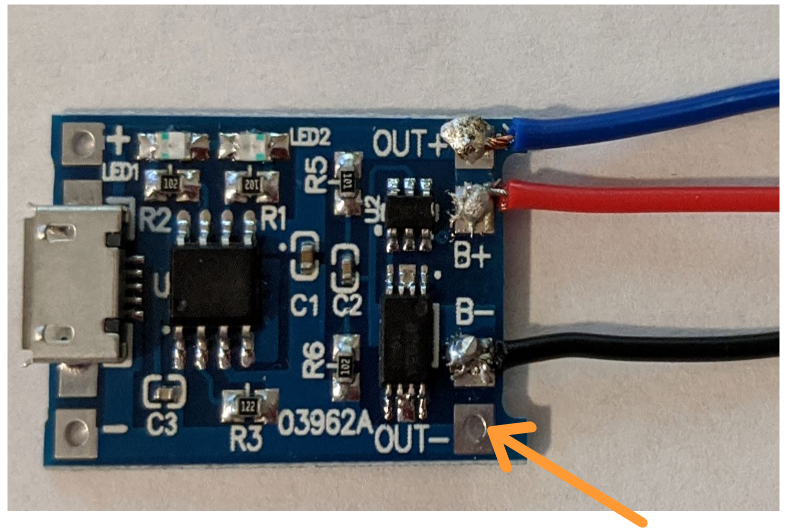
Watch Assembly

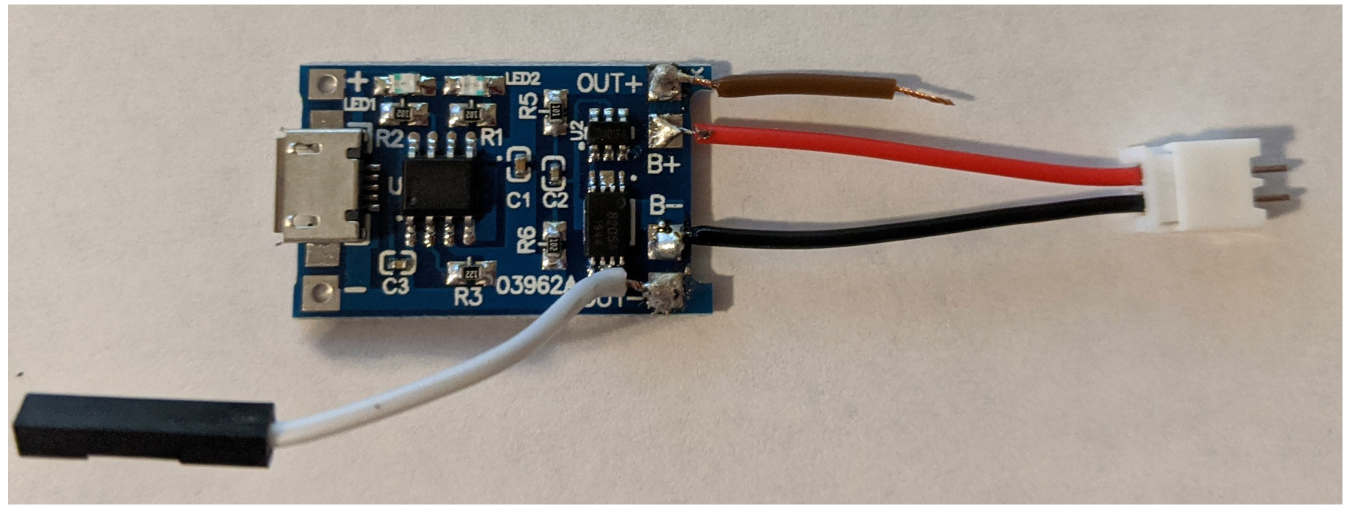




Next, we will solder a small wire with both end stripped to the voltage out positive plated-through hole.

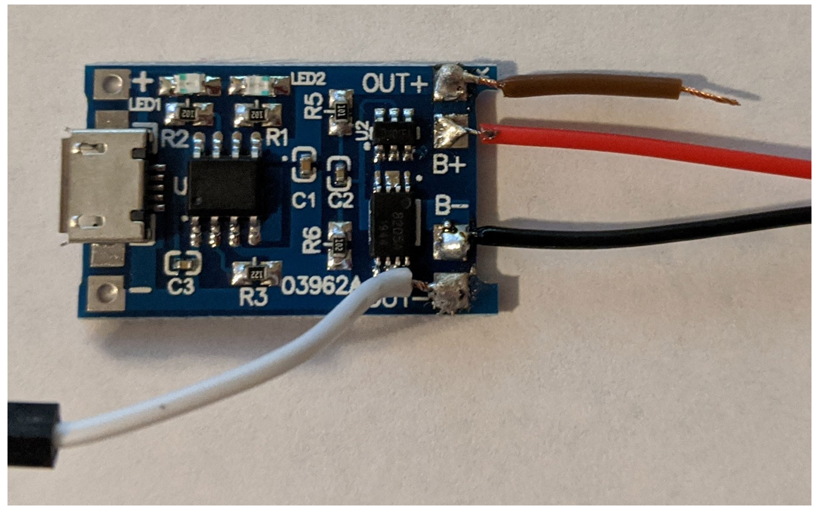
Watch Assembly

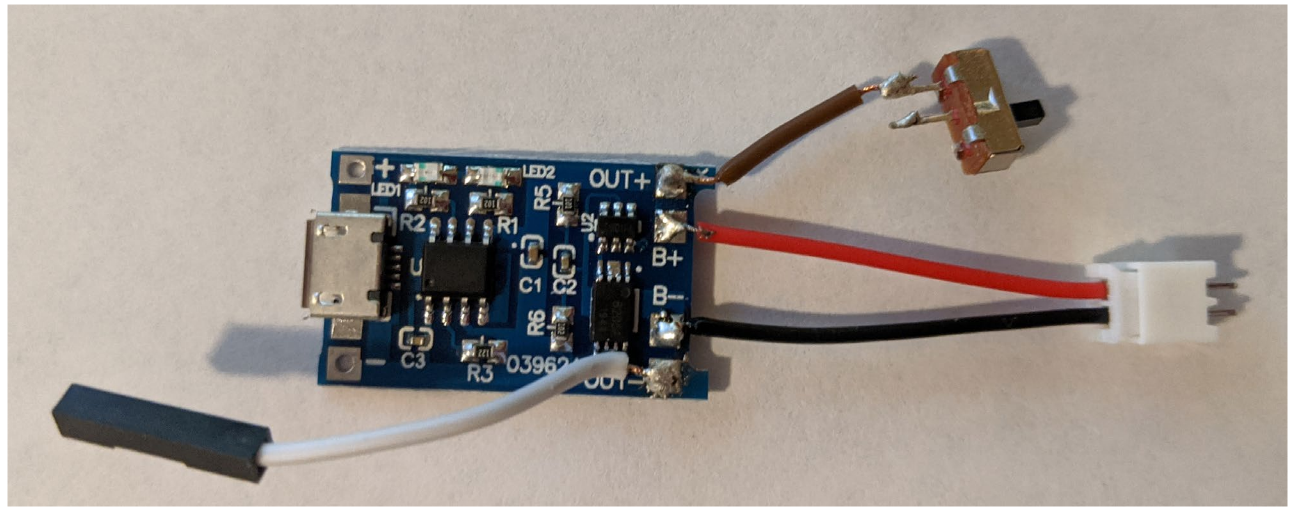




Next, we will solder a jumper wire with a female header housing on the end to the voltage out negative plated-through hole.

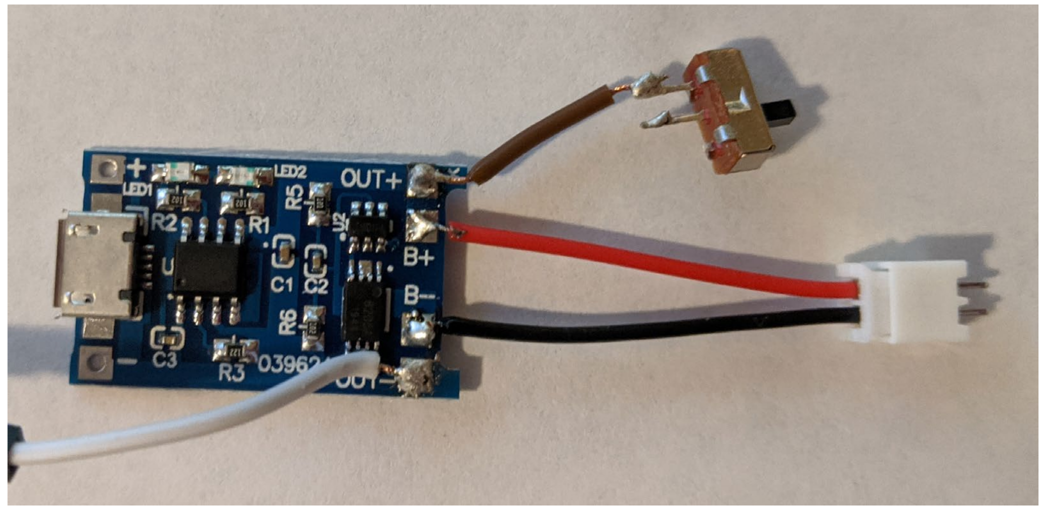
Watch Assembly

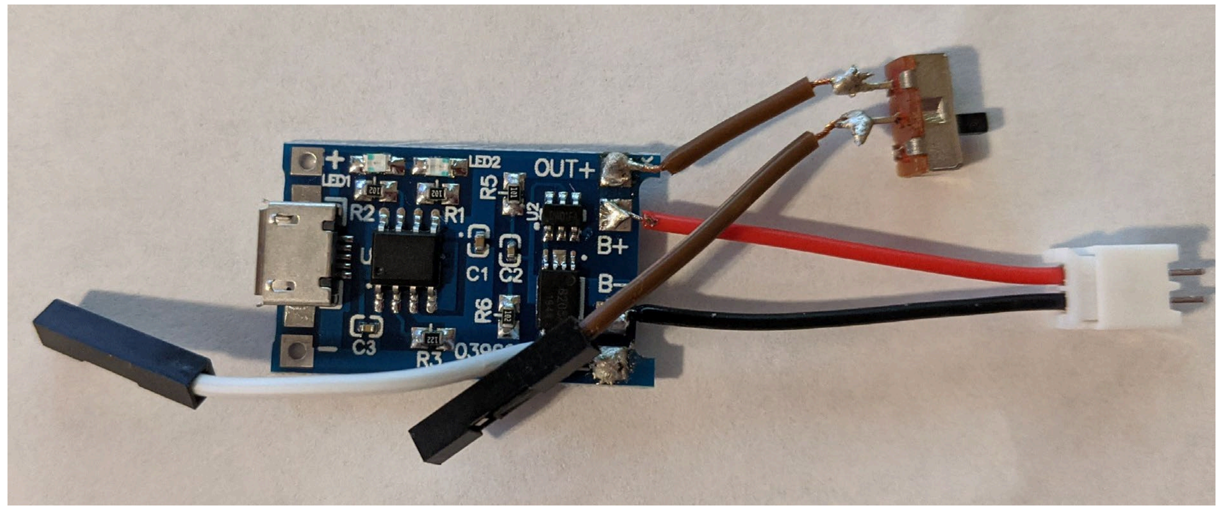




Once the jumper wire is soldered to the negative end of the voltage out, we will then solder the stripped end of the wire connected to the positive end voltage out to the outer pin of the SPDT switch.

Watch Assembly



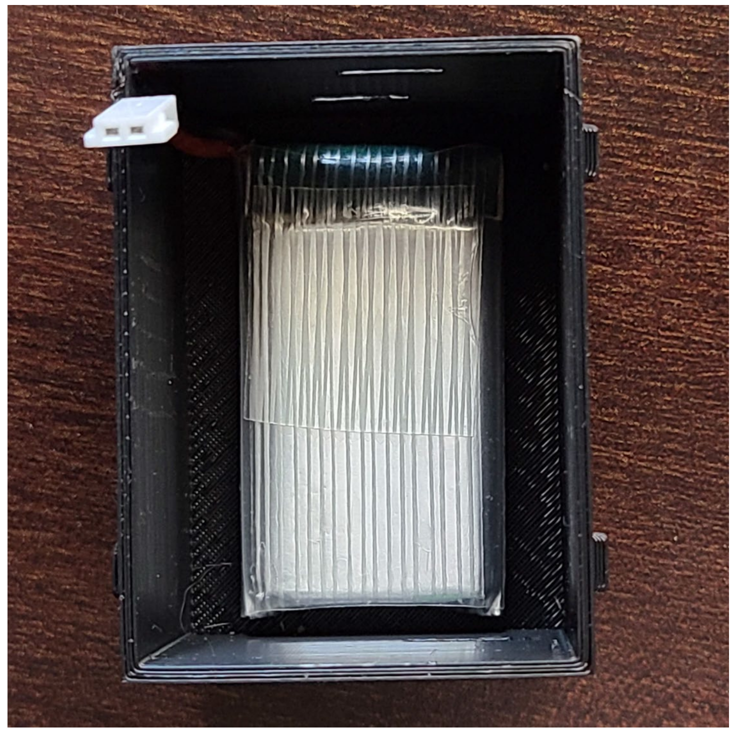


Next, we will solder a jumper wire with a female header housing to the middle pin of the SPDT switch. Once it is soldered to prevent any short circuits, wrap electrical tape around the switch pins.

Watch Assembly

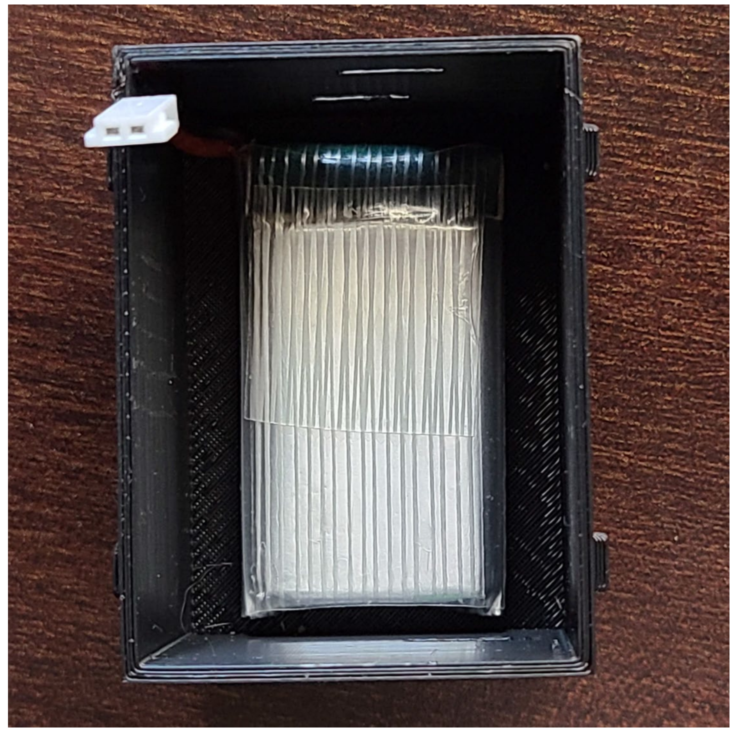
Assembling Components into Watch Case

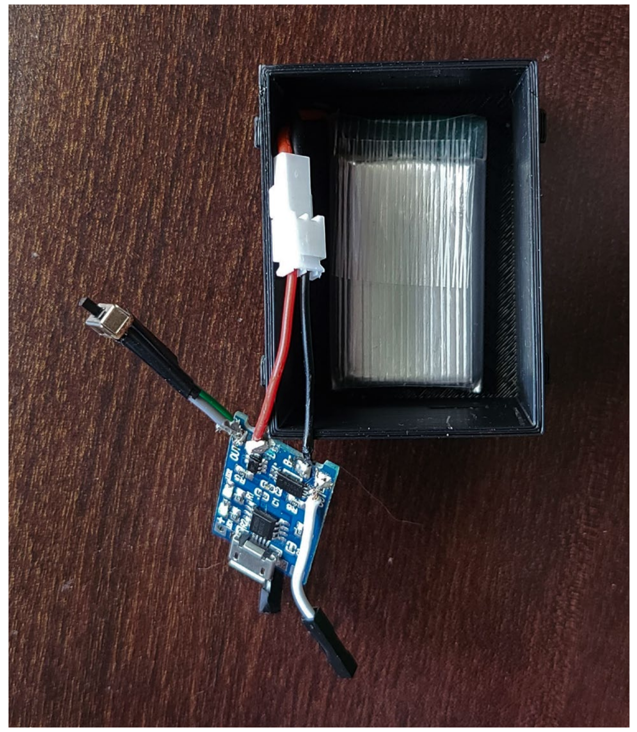
→



To begin assembling all the components into the case, we'll start by placing in the battery. Apply double sided tape to one side of the battery to help keep it secure and place it in the center of the case.

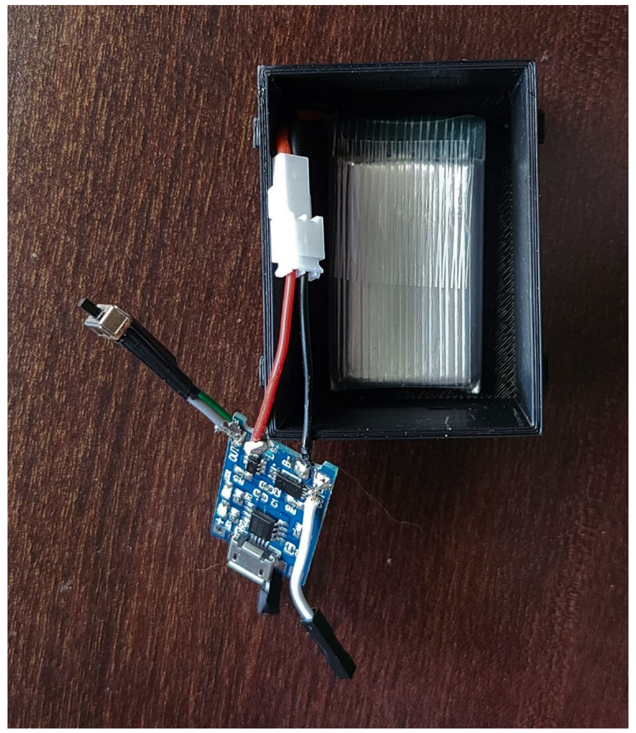
Watch Assembly



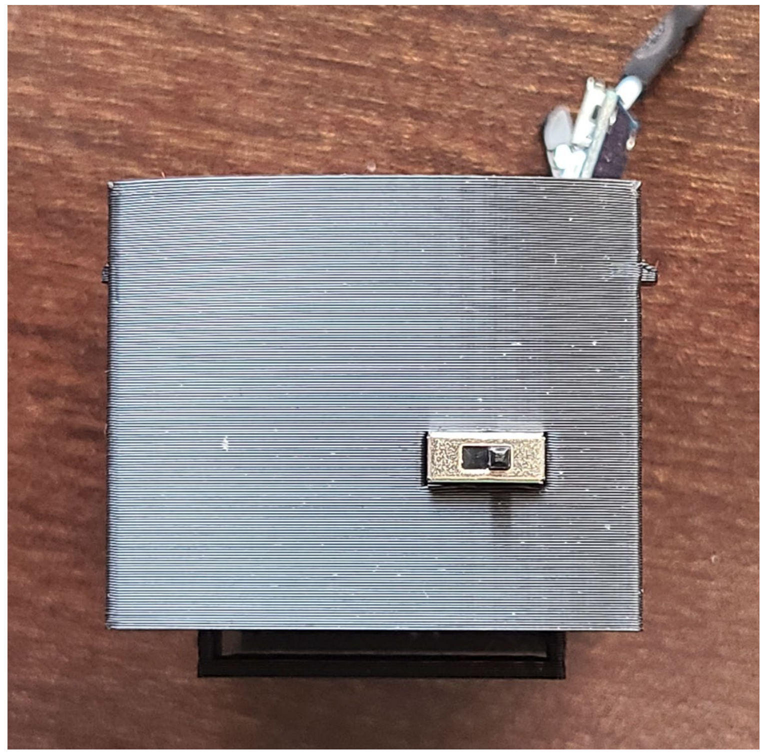


After placing the battery we will connect our charging circuit. Make sure power the is switched off, also that the polarities match to prevent damage to the board and battery! The switch off position is the side opposite to the pin connected to the positive voltage out. For the polarities make sure that red is connected to red, and black is connected to black.

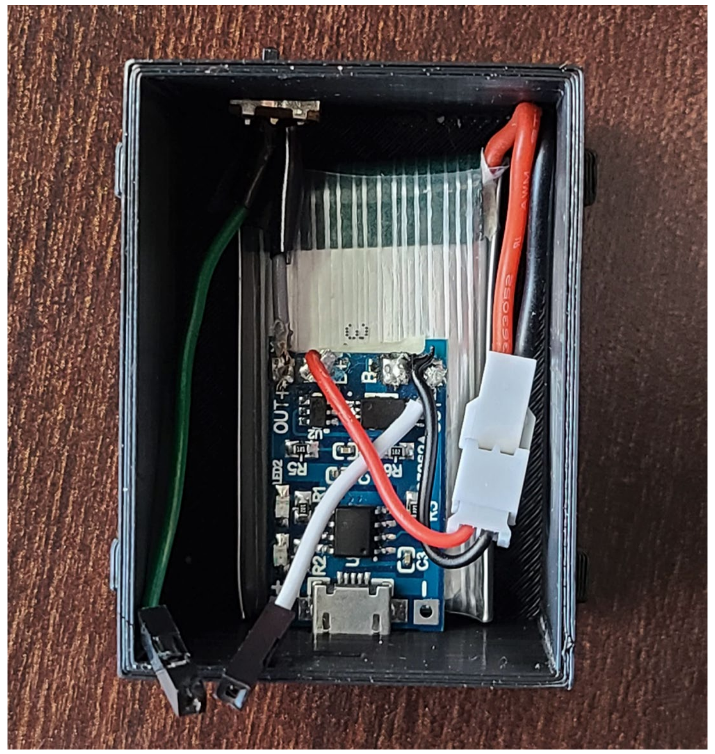
Watch Assembly



→

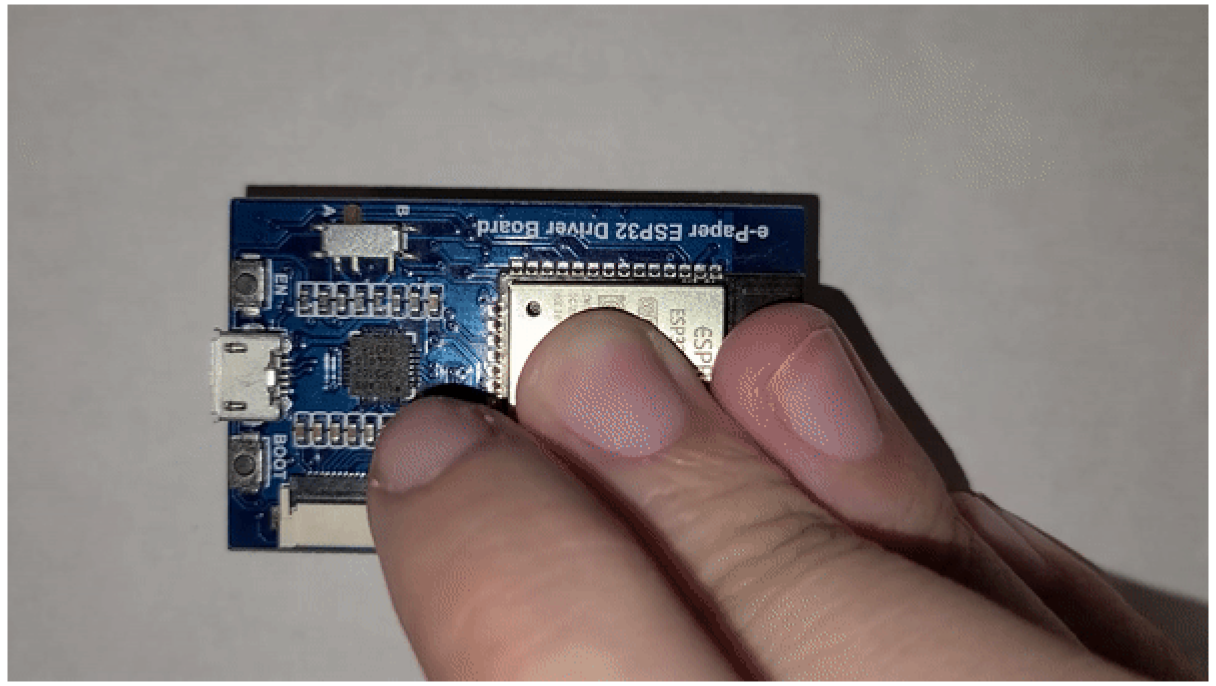


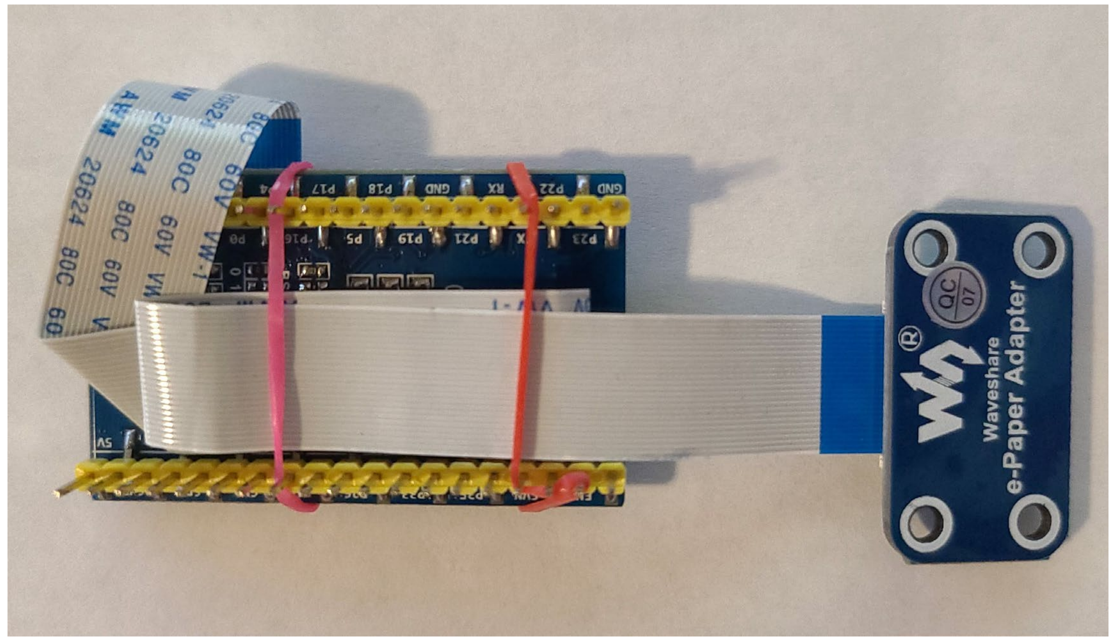
→



Once the battery is connected to the board, take the switch and insert through the cut out. Once the switch is in place, apply a small piece of double-sided tape to the bottom of the board. Next, place the board on the battery making sure that the micro UBS port aligns with the cut out.

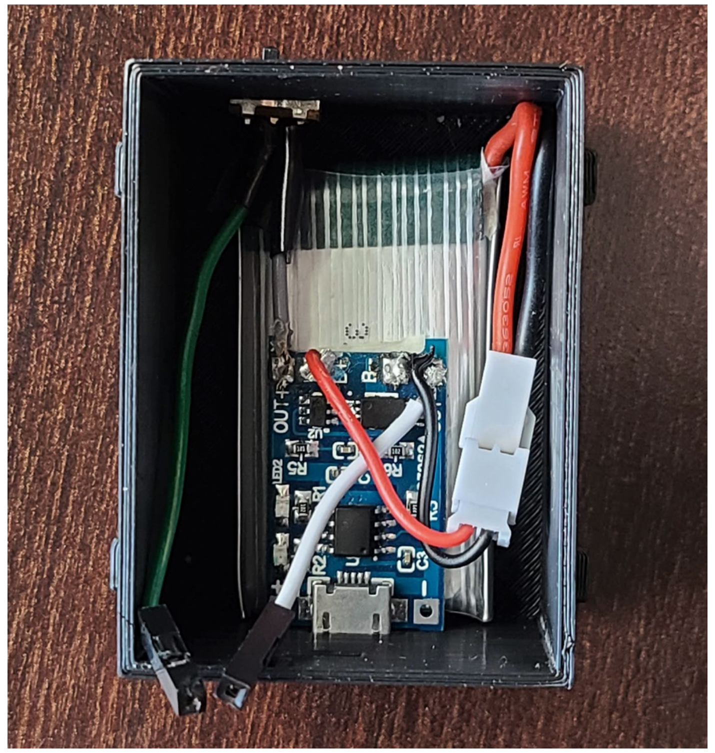
Watch Assembly



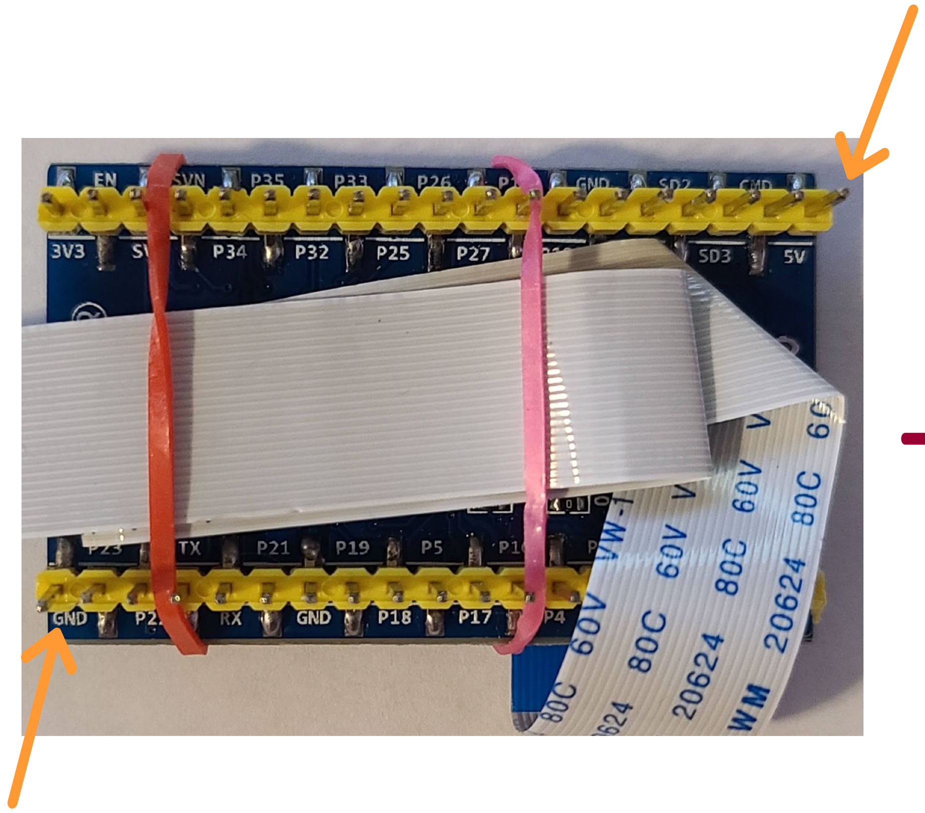


Next, we will connect the SPI extension header to the board. After it is connected, tuck and fold the excess ribbon underneath the board and use small elastic bands to hold it in place. This will make it easier to put the board in the case.

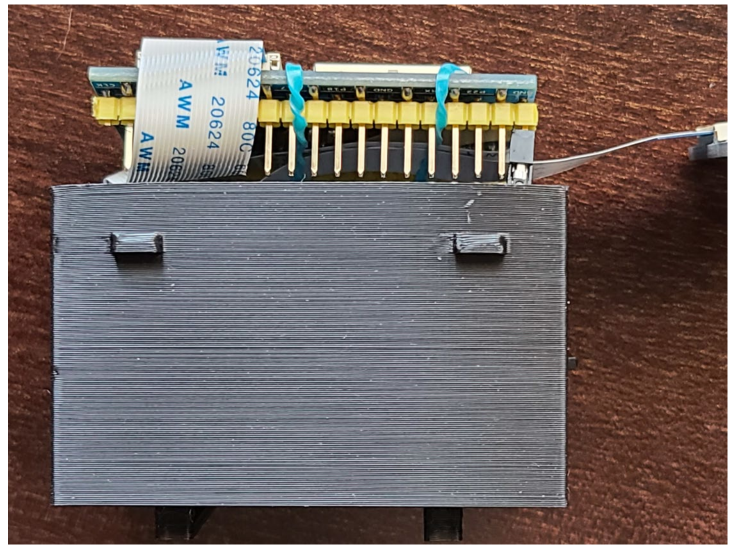
Watch Assembly



→

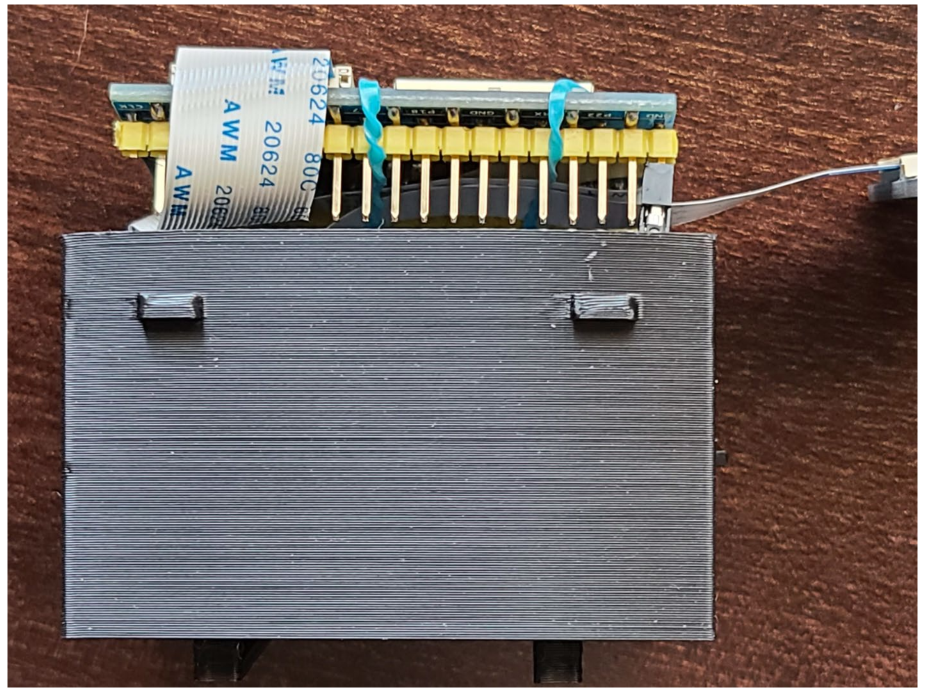


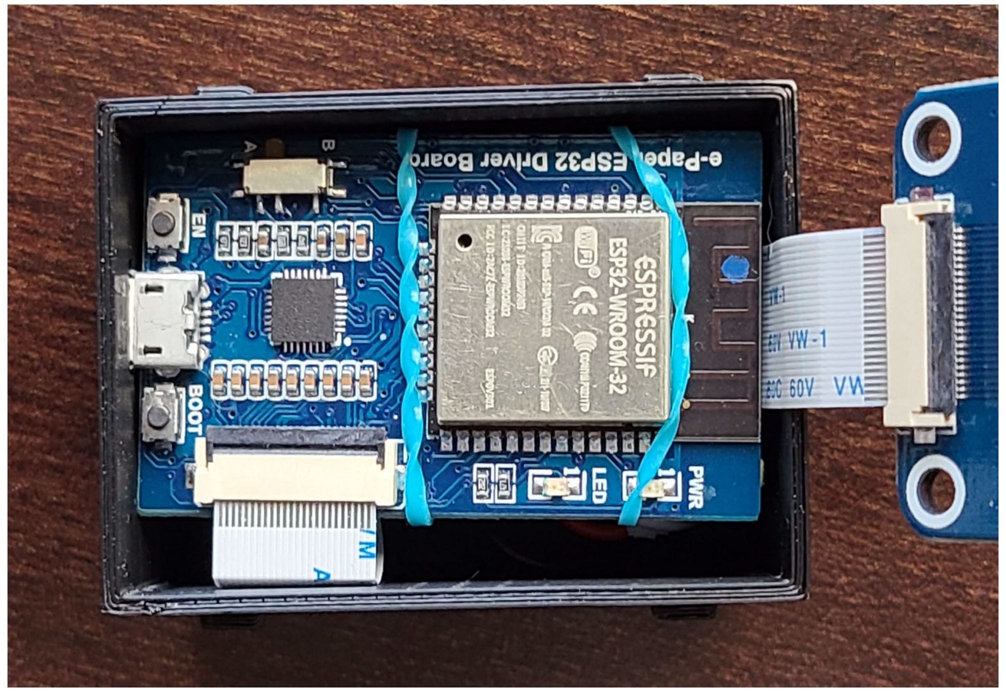
→



Now we will connect the charging board to the driver board. We will first take the jumper wire connected to negative voltage out and plug it into the IO pin labeled GND on the bottom of the board. Next, we will plug in the jumper wire connected to the switch to the IO pin labeled 5V. Make sure these wires are plugged into the correct IO pins to prevent damaged to both boards.

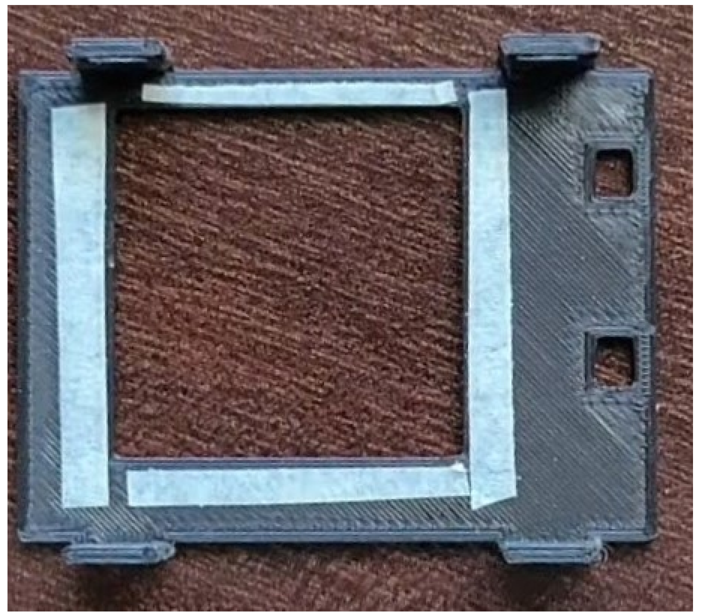
Watch Assembly



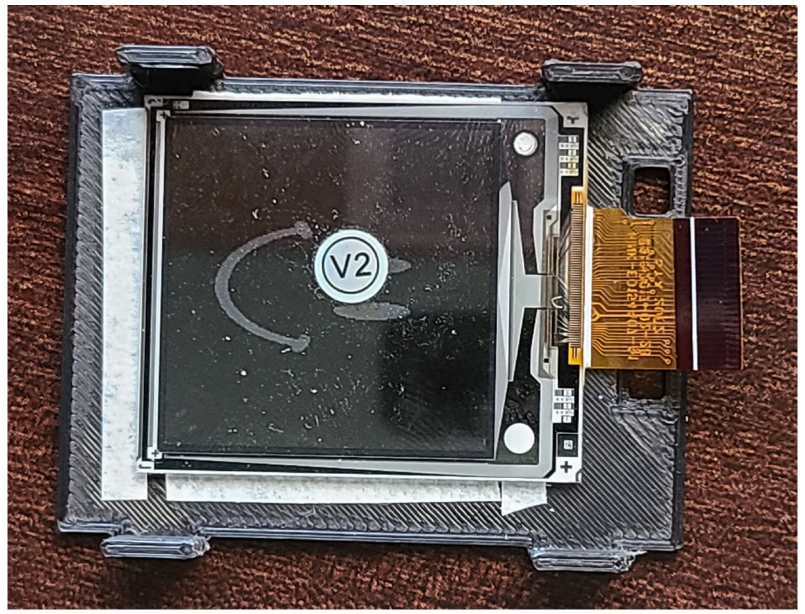


Once the driver board is connected to the charging board, lower the driver board into the case, making sure that the micro USB port lines up with the cut out.

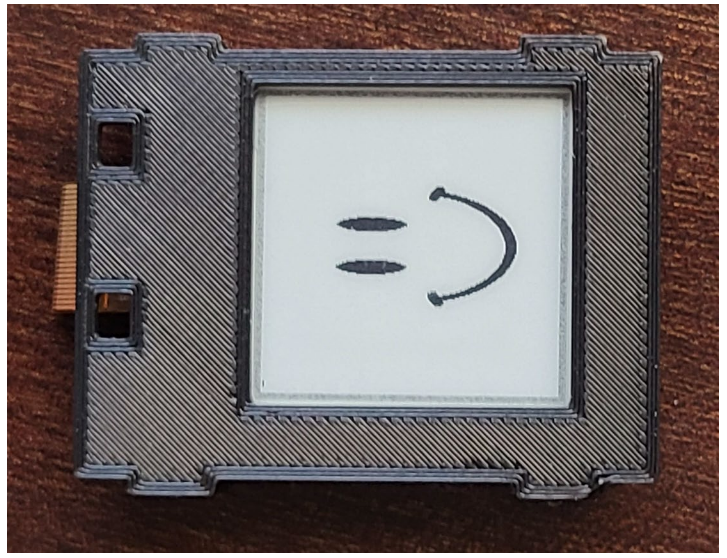
Watch Assembly



→

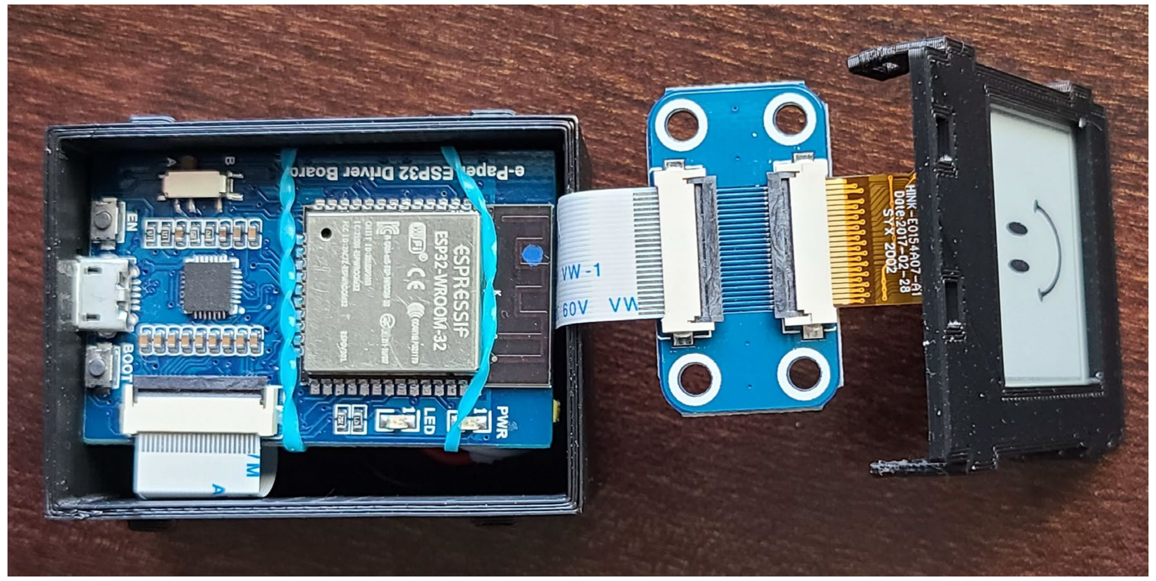


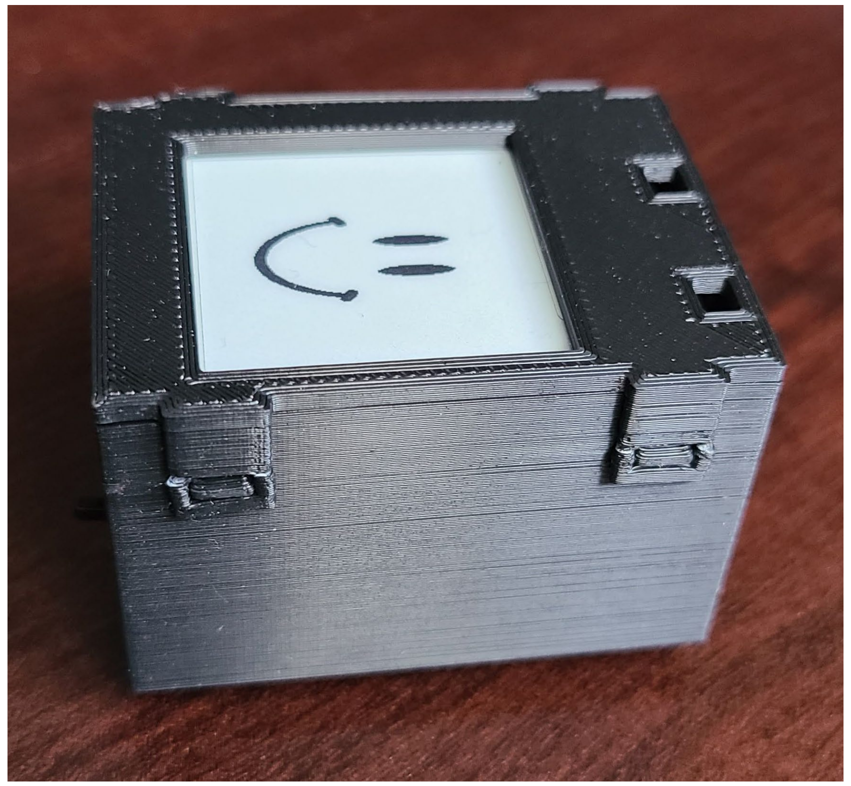
→



Next, apply double-sided tape around the edges of the screen cut out on the top plate. After applying the tape, line up the screen with the cut out and press it into place.

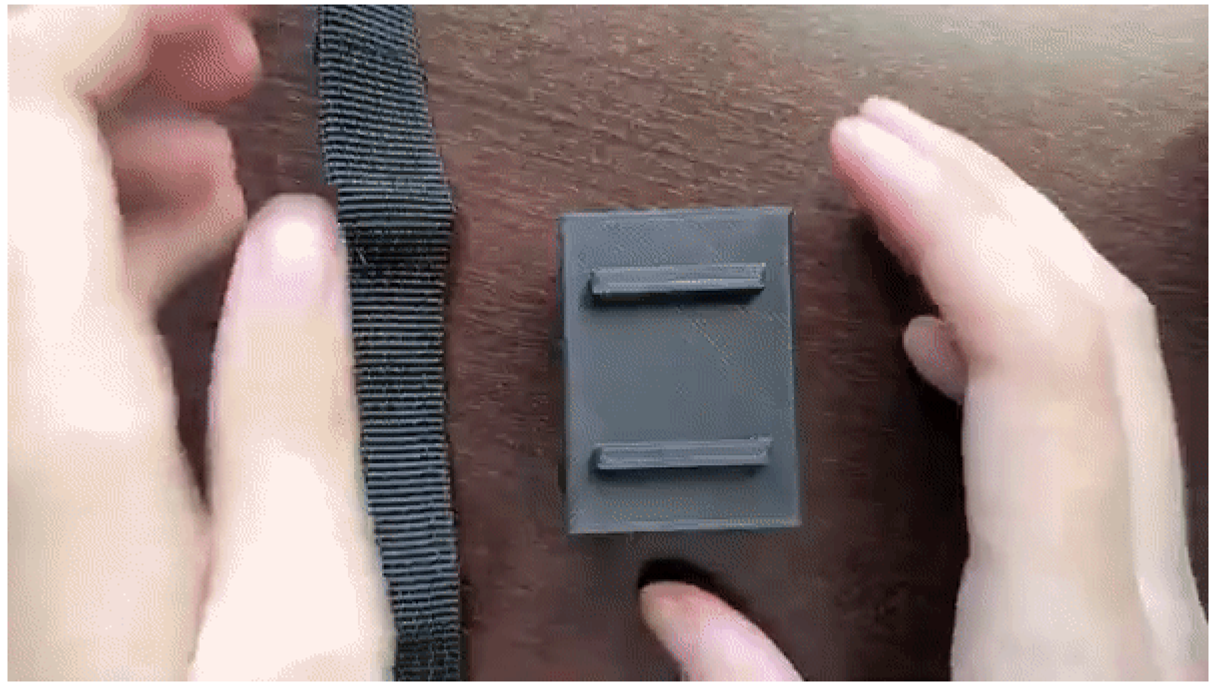
Watch Assembly





Now connect the e-paper screen to the extension header. After, take the top plate and click it into place on the case, making sure the cut outs line up with the buttons on the board.

Watch Assembly



For the final step, take the Velcro strap and thread it through the slits on the bottom of the case.

You have now completed your watch!

WARNING

POWER FROM THE BATTERY MUST BE SWITCHED OFF BEFORE CONNECTING THE ASSEMBLED WATCH TO A COMPUTER!

HAVING POWER FROM BOTH THE USB PORT AND BATTERY COULD RESULT IN DAMAGE TO THE BOARD, BATTERY, AND YOUR COMPUTER!

Time to Reflect!

7. Did you encounter any struggles while assembling the watch? If so, how did you overcome them?

8. What are some of the skills you used in this project that you would like to work to improve?

References

1. 2020. *Universal E-Paper Raw Panel Driver Board*. [image] Available at: <https://www.waveshare.com/e-paper-esp32-driver-board.htm> [Accessed 30 June 2020].

2. Rouse, M., 2020. *What Is A Microcontroller And How Does It Work?*. [online] IoT Agenda. Available at: <https://internetofthingsagenda.techtarget.com/definition/microcontroller> [Accessed 30 June 2020].

3. Rouse, M., 2020. *What Is An Embedded System?*. [online] IoT Agenda. Available at: <https://internetofthingsagenda.techtarget.com/definition/embedded-system> [Accessed 30 June 2020].

4. M, S., 2020. *Arduino - SPI*. [online] Arduino.cc. Available at: <https://www.arduino.cc/en/reference/SPI> [Accessed 30 June 2020].

5. Mitchell, B., 2020. *What Is A USB Port And How Can You Use It?*. [online] Lifewire. Available at: <https://www.lifewire.com/what-is-a-usb-port-818166> [Accessed 30 June 2020].

6. Basics, C., 2020. *Basics Of UART Communication*. [online] Circuit Basics. Available at: <https://www.circuitbasics.com/basics-uart-communication/> [Accessed 30 June 2020].

7. Cs.uwaterloo.ca. 2020. *I/O Ports And Connectors*. [online] Available at: <https://cs.uwaterloo.ca/~brecht/servers/docs/PowerEdge-2600/en/Pe350/UG/1D751ab0.pdf> [Accessed 30 June 2020].

8. 2020. *200X200, 1.54Inch E-Ink Raw Display Panel*. [image] Available at: <https://www.waveshare.com/1.54inch-e-Paper.htm> [Accessed 30 June 2020].

9. 2020. *Appearance Of Pixels*. [image] Available at: <https://en.wikipedia.org/wiki/Electronic_paper> [Accessed 30 June 2020].

10. Bonsor, K., 2020. *How Electronic Ink Works*. [online] HowStuffWorks. Available at: <https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/e-ink1.htm> [Accessed 30 June 2020].

References

11. 2020. *Micro USB 5V 1A 18650 TC4056A Replace TP4056 Lithium Battery Charger Module Charging Board With Dual Functions*. [image] Available at: <https://www.aliexpress.com/i/32216806621.html> [Accessed 3 August 2020].

12. Best Microcontroller Projects. 2020. *The TP4056: Lithium Ion/Polymer Battery Charger IC*. [online] Available at: <https://www.best-microcontroller-projects.com/tp4056.html> [Accessed 3 August 2020].

13. AddOhms, 2014. *Mosfets And How To Use Them | Addohms #11*. [video] Available at: <https://www.youtube.com/watch?v=GrvvkYTW_0k> [Accessed 6 August 2020].

14. 2020. *Salt Water Car Kit*.

15. 3D Hubs. n.d. *What Is 3D Printing? The Definitive Guide | 3D Hubs*. [online] Available at: <https://www.3dhubs.com/guides/3d-printing/> [Accessed 21 July 2020].

16. Dynomotion, n.d. *Gcode Screen*. [image] Available at: <https://www.dynomotion.com/Help/GCodeScreen/GCodeScreen.htm> [Accessed 20 July 2020].

17. ALL3DP, 2020. *G-Code Commands #8: G2 Or "Clockwise Motion"*. [image] Available at: <https://all3dp.com/g-code-tutorial-3d-printer-gcode-commands/> [Accessed 3 August 2020].

18. Cyant. 2017. *Fused Deposition Modeling (FDM) — Cyant*. [online] Available at: <https://www.cyant.co/lexicon/2017/8/20/fused-deposition-modeling-fdm> [Accessed 17 July 2020].

19. Calcetero, M., 2018. *Smart Friday With Robotlab -Computer Science Vs Computer Programming Differences*. [image] Available at: <https://www.robotlab.com/blog/smart-friday-computer-science-vs-computer-programming-differences> [Accessed 10 July 2020].

20. Lewis, C., 2019. *Is Coding Over? Why Learning To Code Is Really About Learning To Learn.*. [image] Available at: <https://www.edsurge.com/news/2019-05-02-is-coding-over-why-learning-to-code-is-really-about-learning-to-learn> [Accessed 16 July 2020].

References

21. Arduino.cc. n.d. *Arduino - Software*. [online] Available at: <https://www.arduino.cc/en/main/software> [Accessed 19 July 2020].

22. Arduino, n.d. *Arduino*. [image] Available at: <http://www.arduino.org/> [Accessed 19 July 2020].

23. Arduino.cc. n.d. *Arduino - Sketch*. [online] Available at: <https://www.arduino.cc/en/tutorial/sketch> [Accessed 19 July 2020].

24. Arduino.cc. n.d. *Arduino - Libraries*. [online] Available at: <https://www.arduino.cc/en/main/libraries> [Accessed 19 July 2020].

25. GeeksforGeeks. n.d. *Functions In C/C++ - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/functions-in-c/> [Accessed 9 August 2020].

26. JournalDev. n.d. *Functions In C Programming - Journaldev*. [online] Available at: <https://www.journaldev.com/30509/functions-in-c-programming> [Accessed 9 August 2020].

27. Tutorialspoint.com. n.d. *C - Functions - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/cprogramming/c_functions.htm> [Accessed 9 August 2020].

28. Edureka. 2019. *Functions In C Programming | C Fundamentals | Edureka*. [online] Available at: <https://www.edureka.co/blog/functions-in-c/> [Accessed 9 August 2020].

29. Tutorials Point, n.d. *Defining A Function*. [image] Available at: <https://www.tutorialspoint.com/cprogramming/c_functions.htm> [Accessed 9 August 2020].

30. Programiz.com. n.d. *C Struct (Structures)*. [online] Available at: <https://www.programiz.com/c-programming/c-structures> [Accessed 9 August 2020].

Thank you to RBC for Sponsoring WiTT Curriculum Kits

RBC

RBC is supporting all of Mohawk College’s Women in Technology and Trades initiatives as part of their Future Launch Program

You have completed this kit!

Please see our site for more curriculum kits and other content.

https://www.mohawkcollege.ca/about-mohawk/cyber-security/science-technology-engineering-and-math-stem-learning-resources