User Guide Standards

Arduino driving board

Table of Contents

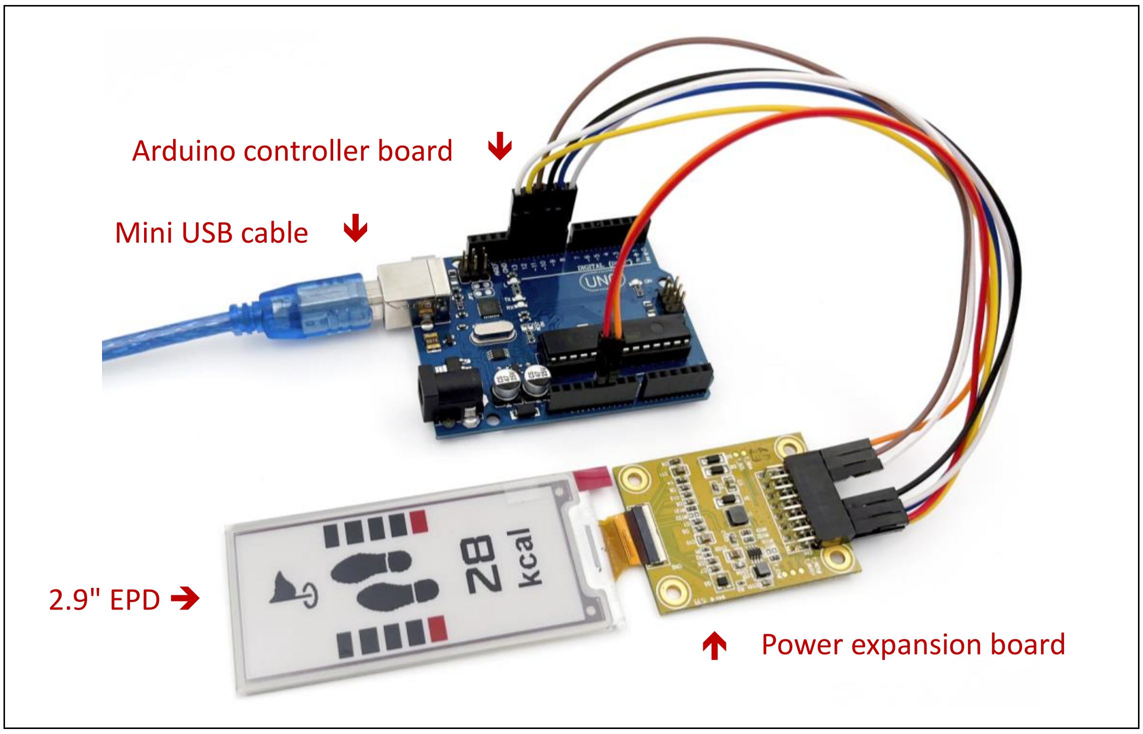
Firmware Update SOP

1 Overview

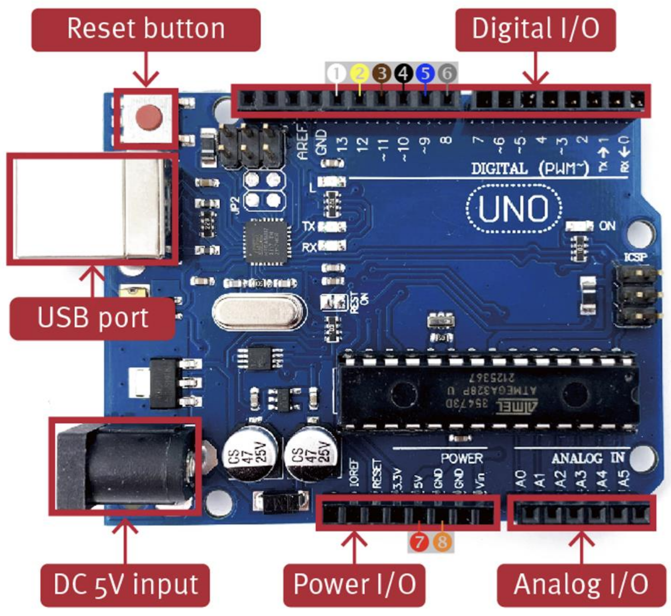
This document specifies a reference design for a hardware timing controller (TCON) supporting the E Ink EA2220-BJC, a 2.9” display. The hardware TCON provides control signals for the source driver and gate drivers in order to drive the display properly. E Ink’s hardware TCON is high performance and easy to integrate with common MCUs and allows users to rapidly design systems using E Ink displays. Features include suitable for Electronic Shelf Label applications.

2 Features

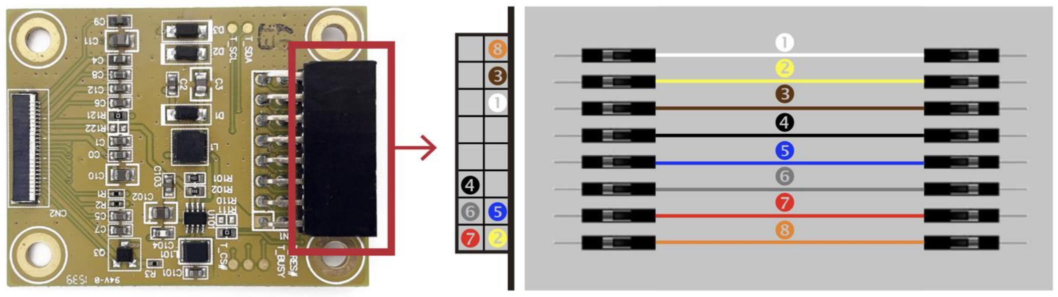
- All In One Hardware EPD Timing Controller for EA2220-BJC
- SPI Serial interface to EPD display
- 128x296 resolution support
- 3 color support (Black and White and Red)
- Waveform mode support
- GC (global cleaning) mode (Black and White and Red)



3 Block Diagram



Pin# on Arduino	Wire number/ color
13	1 /White
12	2 /Yellow
11	3 /Brown
10	4 /Black
9	5 /Blue
8	6 /Gray
5V	7 /Red
GND	8 /Orange



* The purple and green Dupont cable are reserved for backup

Follow the pin connection between Arduino controller board and power expansion board to wire the Dupont cable number by number, which shows as the diagram above.

E Ink EA2220-BJC 2.9” Panel

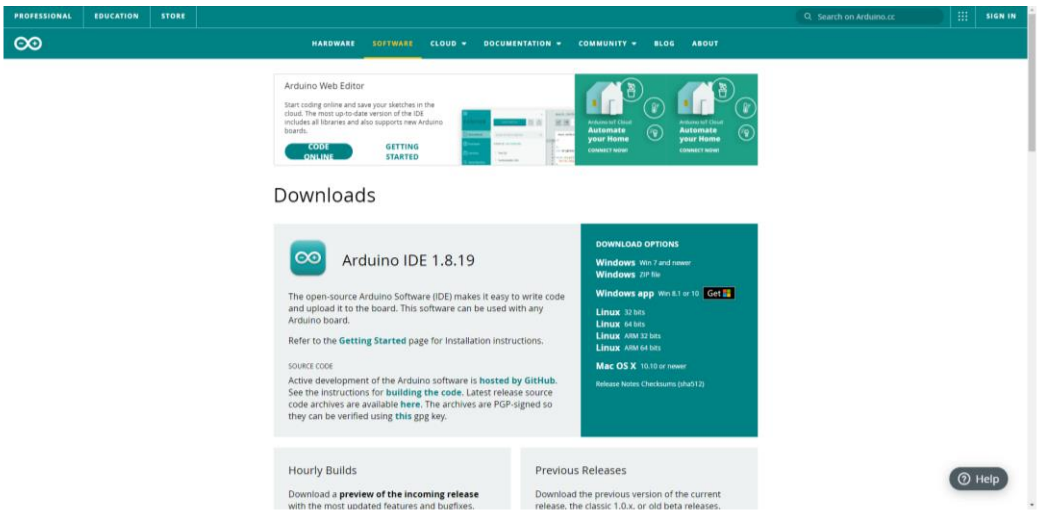
E Ink’s EA2220-BJC (2.9”) panel is an active matrix reflective electrophoretic display module using advanced plastic substrate technology. The display has a resolution of 128x296 pixels and has the ability to reproduce red in addition to black and white. The plastic substrate is protected by an outer covering, serving as environmental and physical protection, which is integrated on the display.

4 Application Programming Interface

The hardware TCON design flow and requirements are listed here for the EA2220-BJC 2.9” EPD display.

(1) Install Arduino Compiler

- Please refer to https://www.arduino.cc/en/Main/Software to install Arduino Compiler, for example Install to d:\arduino.



Arduino Web Editor

Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

Downloads

Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer
Release Notes Checksums (sha512)

Hourly Builds

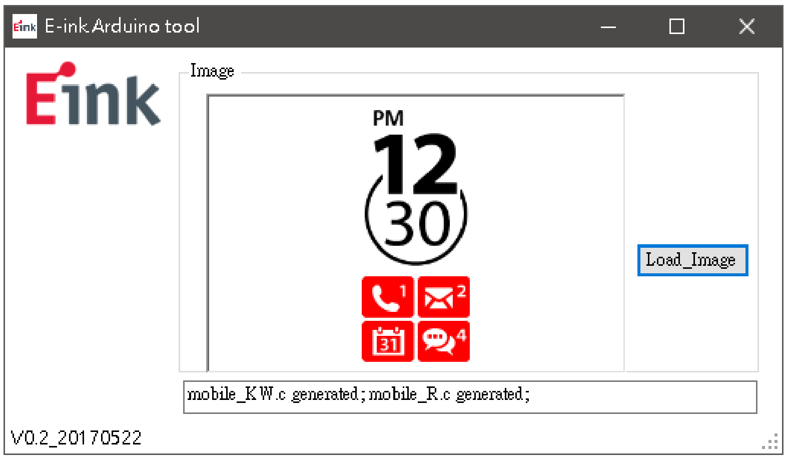
Download a **preview of the incoming release** with the most updated features and bugfixes.

Previous Releases

Download the previous version of the current release, the classic 1.0.x, or old beta releases.

(2) Convert image to header files

Please run **Arduino_tool.exe** and use [Load_Image] button to select an image with resolution 128x296, corresponding [image file name]_KW.h & [image file name]_R.h will be generated.

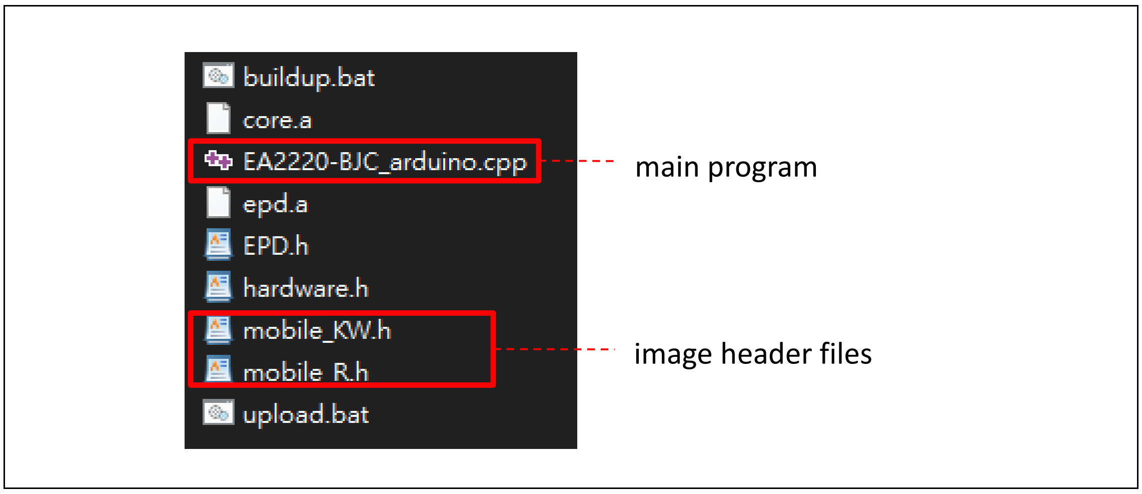


```
mobile_KW.c generated; mobile_R.c generated;
```

As this example, you would get <u>mobile_KW.h</u> and <u>mobile_R.h</u> in the same folder for following compilation.

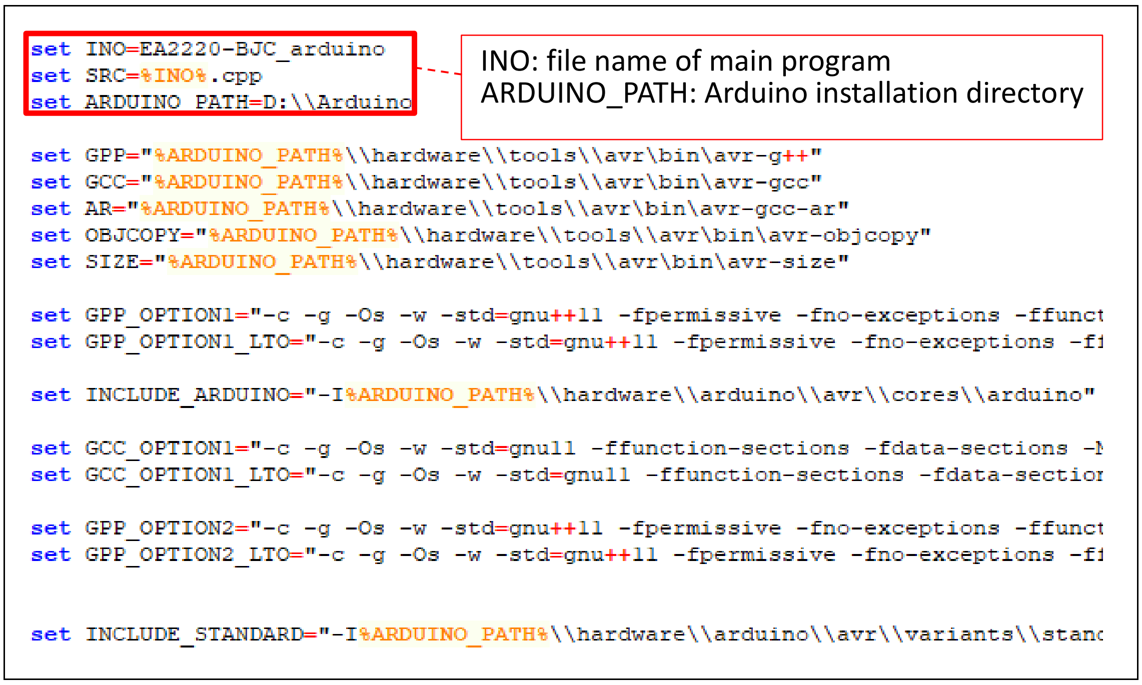
(3) Create your own code

- 1. Unzip eink029tr3_arduino.zip, for example to d:\,
  there will be a folder d:\EA2220-BJC_arduino
- 2. Double click the EA2220-BJC_arduino.cpp file as sample code to create your own code. Check chapter 5 for example code structure.
- 3. Drag the image header files to this folder directory



(4) Compile and upload to Arduino

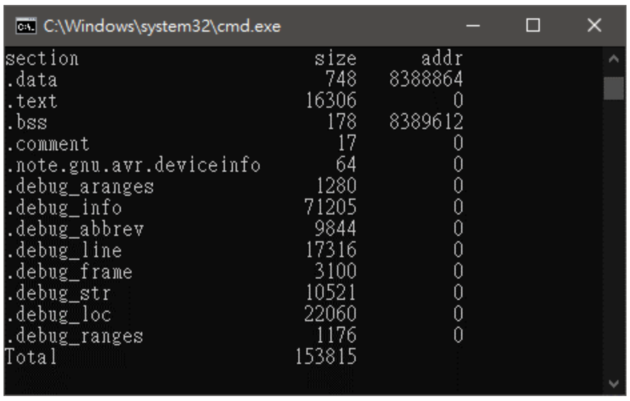
- Use text editor to open buildup.bat. According to the Arduino installation directory, copy this directory and paste to replace the value of “ARDUINO_PATH”. The value of “INO” should be the name of main program(.cpp).



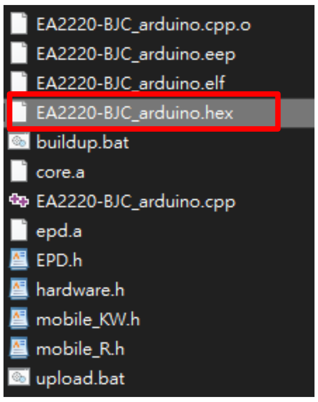
```
set INO=EA2220-BJC_arduino
set SRC=%INO%.cpp
set ARDUINO_PATH=D:\\Arduino

set GPP="%ARDUINO_PATH%\\hardware\\tools\\avr\bin\avr-g++"
set GCC="%ARDUINO_PATH%\\hardware\\tools\\avr\bin\avr-gcc"
set AR="%ARDUINO_PATH%\\hardware\\tools\\avr\bin\avr-gcc-ar"
set OBJCOPY="%ARDUINO_PATH%\\hardware\\tools\\avr\bin\avr-objcopy"
set SIZE="%ARDUINO_PATH%\\hardware\\tools\\avr\bin\avr-size"

set GPP_OPTION1="-c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunct
set GPP_OPTION1_LTO="-c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -fi

set INCLUDE_ARDUINO="-I%ARDUINO_PATH%\\hardware\\arduino\\avr\\cores\\arduino"

set GCC_OPTION1="-c -g -Os -w -std=gnu11 -ffunction-sections -fdata-sections -M
set GCC_OPTION1_LTO="-c -g -Os -w -std=gnu11 -ffunction-sections -fdata-sectior

set GPP_OPTION2="-c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunct
set GPP_OPTION2_LTO="-c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -fi


set INCLUDE_STANDARD="-I%ARDUINO_PATH%\\hardware\\arduino\\avr\\variants\\stanc
```

INO: file name of main program
ARDUINO_PATH: Arduino installation directory

● Execute this buildup.bat, the building result information will show in the end and the .hex file will be generated.

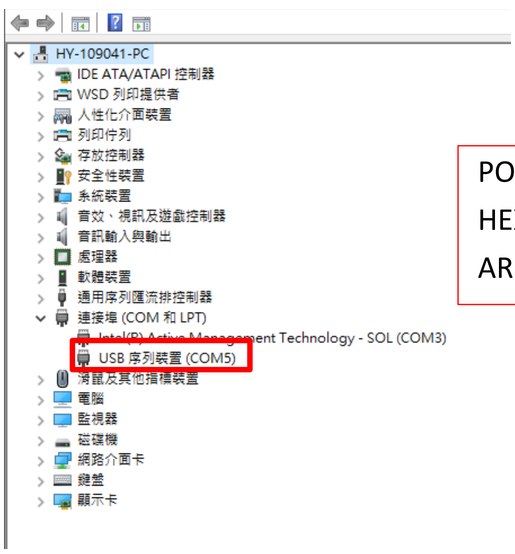


```
section                      size      addr
.data                         748   8388864
.text                       16306         0
.bss                          178   8389612
.comment                       17         0
.note.gnu.avr.deviceinfo       64         0
.debug_aranges               1280         0
.debug_info                 71205         0
.debug_abbrev                9844         0
.debug_line                 17316         0
.debug_frame                 3100         0
.debug_str                  10521         0
.debug_loc                  22060         0
.debug_ranges                1176         0
Total                      153815
```

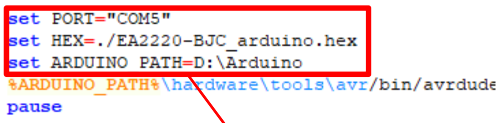


```
EA2220-BJC_arduino.cpp.o
EA2220-BJC_arduino.eep
EA2220-BJC_arduino.elf
EA2220-BJC_arduino.hex
buildup.bat
core.a
EA2220-BJC_arduino.cpp
epd.a
EPD.h
hardware.h
mobile_KW.h
mobile_R.h
upload.bat
```

● Use text editor to open upload.bat (as lower right figure). After connecting Arduino board to PC through USB, check the COM port assigned by device manager (as lower left figure)and modify the value of "PORT" in upload.bat. The value of "HEX" and "ARDUINO_PATH" both should be confirmed as well.



```
HY-109041-PC
  IDE ATA/ATAPI 控制器
  WSD 列印提供者
  人性化介面裝置
  列印佇列
  存放控制器
  安全性裝置
  系統裝置
  音效、視訊及遊戲控制器
  音訊輸入與輸出
  處理器
  軟體裝置
  通用序列匯流排控制器
  連接埠 (COM 和 LPT)
    Intel(R) Active Management Technology - SOL (COM3)
    USB 序列裝置 (COM5)
  滑鼠及其他指標裝置
  電腦
  監視器
  磁碟機
  網路介面卡
  鍵盤
  顯示卡
```



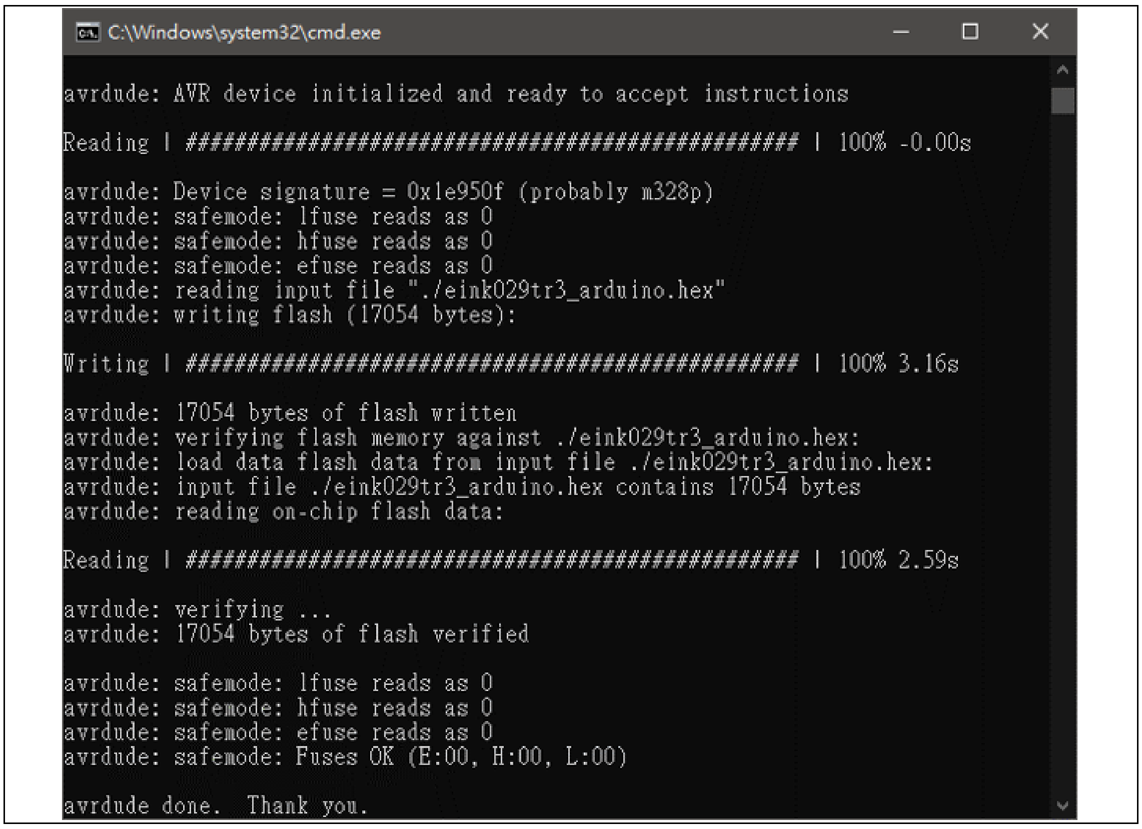
```
set PORT="COM5"
set HEX=./EA2220-BJC_arduino.hex
set ARDUINO_PATH=D:\Arduino
%ARDUINO_PATH%\hardware\tools\avr/bin/avrdude
pause
```

PORT: PC virtual com port

HEX: hex file directory

ARDUINO_PATH: Arduino installation directory

● Execute upload.bat, the uploading information will show in the end and the image will be displaying on EPD.


C:\Windows\system32\cmd.exe

```
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% -0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: safemode: lfuse reads as 0
avrdude: safemode: hfuse reads as 0
avrdude: safemode: efuse reads as 0
avrdude: reading input file "./eink029tr3_arduino.hex"
avrdude: writing flash (17054 bytes):

Writing | ################################################## | 100% 3.16s

avrdude: 17054 bytes of flash written
avrdude: verifying flash memory against ./eink029tr3_arduino.hex:
avrdude: load data flash data from input file ./eink029tr3_arduino.hex:
avrdude: input file ./eink029tr3_arduino.hex contains 17054 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 2.59s

avrdude: verifying ...
avrdude: 17054 bytes of flash verified

avrdude: safemode: lfuse reads as 0
avrdude: safemode: hfuse reads as 0
avrdude: safemode: efuse reads as 0
avrdude: safemode: Fuses OK (E:00, H:00, L:00)

avrdude done.  Thank you.
```

5 Example Codes

(1) Functions

● EPD Initialization

TCON driver initialize, we have to set EPD_Init() before use the TCON Driver API

Item name	Description
Function prototype	void EPD_Init()
Parameter	Null
Return	Null

● EPD display image

To display image pixel data to EPD

Item name	Description
Function prototype	bool EPD_Display_Image(const unsigned char img_kw[], const unsigned char img_r[])
Parameter	Img_kw: Pixel data array for Black and White Img_r: Pixel data array for Red
Return	bool: Time out flag

(2) Sample code

```
#include "EPD.h"
#include "hardware.h"
#include “mobile_KW.h”    //Can be generated by user using Arduino_tool.exe
#include “mobile_R.h”    //Can be generated by user using Arduino_tool.exe

unsigned int case_selection = 0;
void setup()
{
        EPD_Init();    //HW-TCON Initialization
}
void loop()
{
        EPD_Display_image(mobile_KW, mobile_R);    //To display calendar image to EPD
        delay(3000);    //Delay 3000ms to go on next loop
}
void testLed(void)    //Keep this for compilation
{
        while(1)
        {
            delay(100);
            digitalWrite(7, HIGH);
            delay(100);
            digitalWrite(7, LOW);
        }
}
```

6 Waveform

Each temperature LUT contains specific voltage information used to generate high quality images on the display. E Ink displays are image stable, meaning that once an image is produced on the display, power can be removed from the system and the image is retained.　The LUT can contain voltage information to produce gray scale as well as compensate for changes in temperature in the environment.　The LUT assumes that previous and next picture data is available.

Display Update Modes - GC mode

The grayscale clearing (GC) mode is used to update the full display and provide a high image quality. When GC is used with Full Display Update the entire display will update as the new image is written. If a Partial Update command is used the only pixels with changing grayscale values will update. The GC mode has 3 unique gray levels (Black, White and Red).

7 Auto Run Guide

After all essential boards are connected and Arduino is powered on, please follow the steps below to start up:

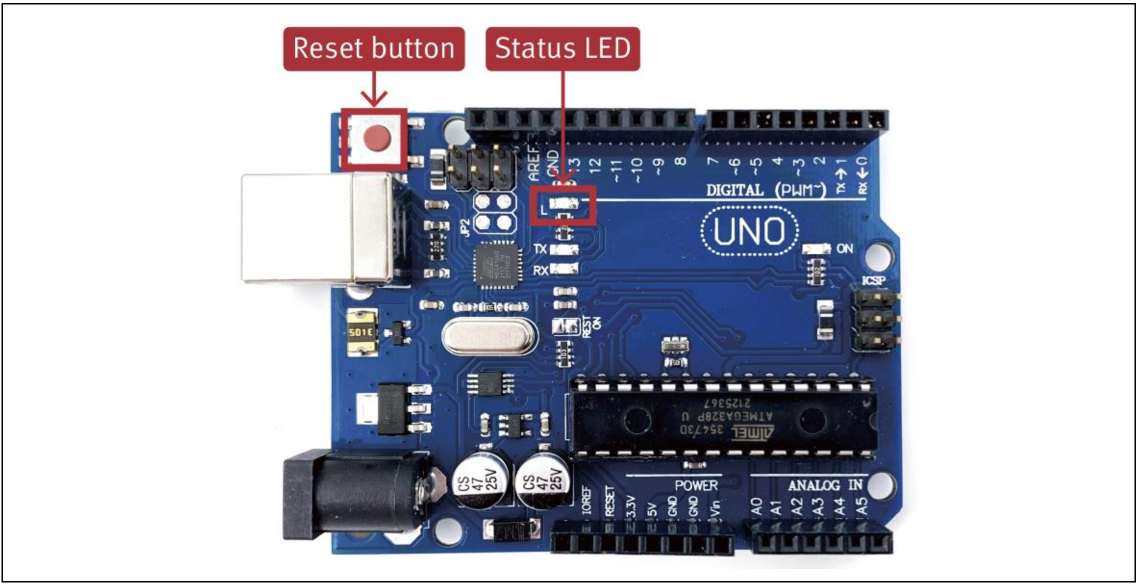
1. Press reset button on Arduino. **(if display process is not activated in 10 seconds, please reset again)**
2. The status LED is on for about 3 seconds then off. (initial process)
3. After display is finished, the status LED is on for 4 seconds then off. (display finish)

Notifications:

1 Do not hot plug the EPD when power is supplying and displaying pattern.

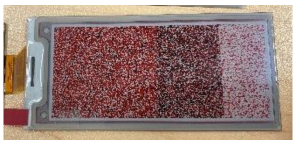
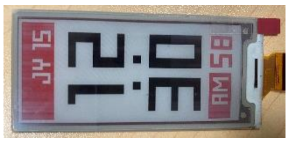
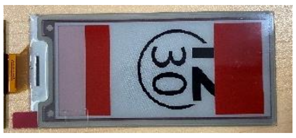
2 To keep the pattern, please plug out power cable when step 3 is completely done.

3 If the displaying behavior is abnormal, please refer to chapter 8 to fix.



8 Trouble shooting

Several abnormal phenomenon are listed as following table, please take the corresponding actions to fix it.

Abnormal situation	Root cause	Action to fix
Noise pattern 	Initial setting failure	Press reset button on Arduino to retry
Color fading 	Charging voltage of source line is lost	Plug out power after the status of ending LED is off to make sure driving is complete
Pattern shifting 	Initial setting failure	Press reset button on Arduino to retry
No response	1.Initial setting failure 2.OTP WF doesn't exist	1.Press reset button on Arduino to retry 2.Contact customer service.

9 Revision History

Version	Date	Page	Description	Author
1.0	2021/12/23		Initial	Vince Lin