LCD Displays – What are they?

LCD – Liquid Crystal Display

Liquid Crystals rotate polarized light – they are more about blocking light than emitting it

https://whatis.techtarget.com/definition/LCD-liquid-crystal-display

Different kinds as technology has evolved

Low power, thin

Considerations - what do you want/need?

How big a display – how much space do you have

Power considerations – current draw, voltage

Interface type – SPI? I2C? Other?

CPU resources needed – RAM? Clock Speed? Storage space? Data pins/address lines?

Cost

Colour/Monochrome

Entry Level Displays - The Good News

2X16 line displays

Can run on 5V

Inexpensive

Have been around a long time.

Tons of examples for just about every language, microcontroller. Lots of programming languages have libraries available to make the project super easy.

No special pins are needed from the microcontroller

Easy to scroll or blink a message too

The Less Than Good News

Takes at least 4, and sometimes 8 data lines in addition to control lines for Enable, Command/Data, possibly Read/Write – 6 lines minimum

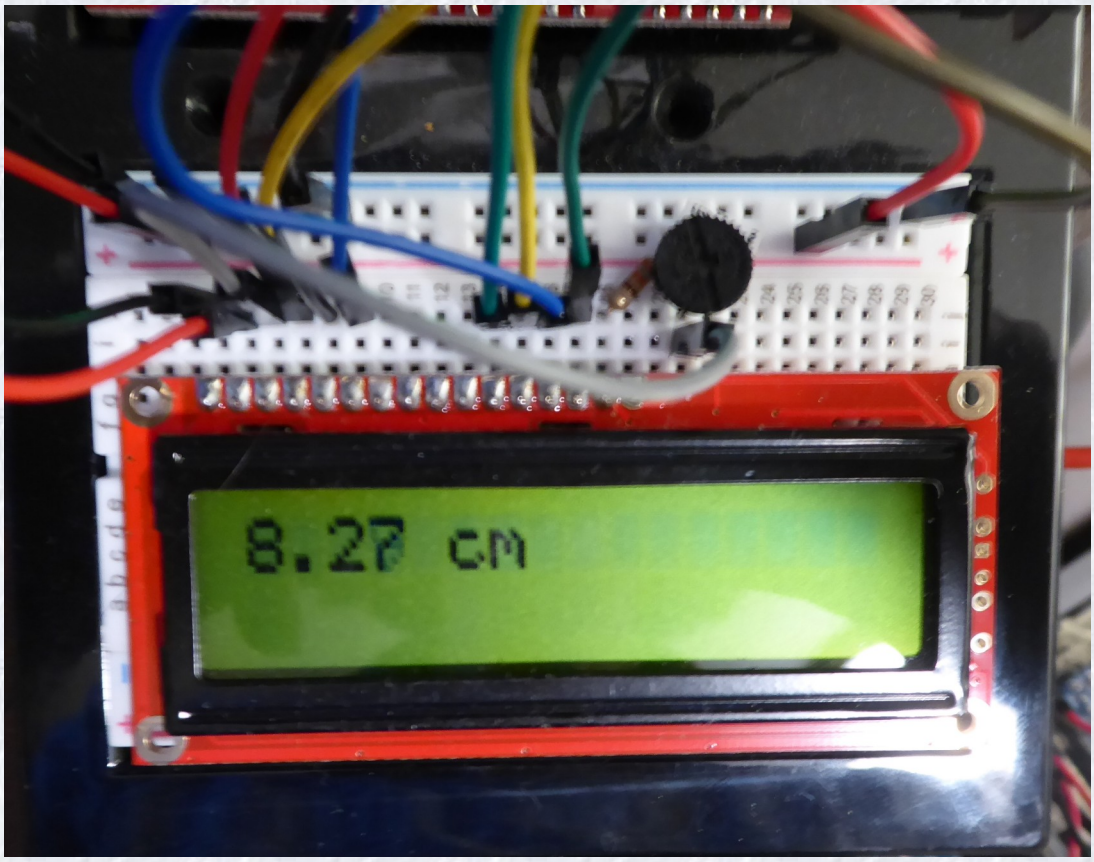
Monochrome only, although some newer ones can change background colours.

Text display of 2 lines and 16 characters, fixed font/size – 16 blocks of 5 px x 5 px grids

Displaying any kind of graphics is challenging

My 16x2 LCD Display Project

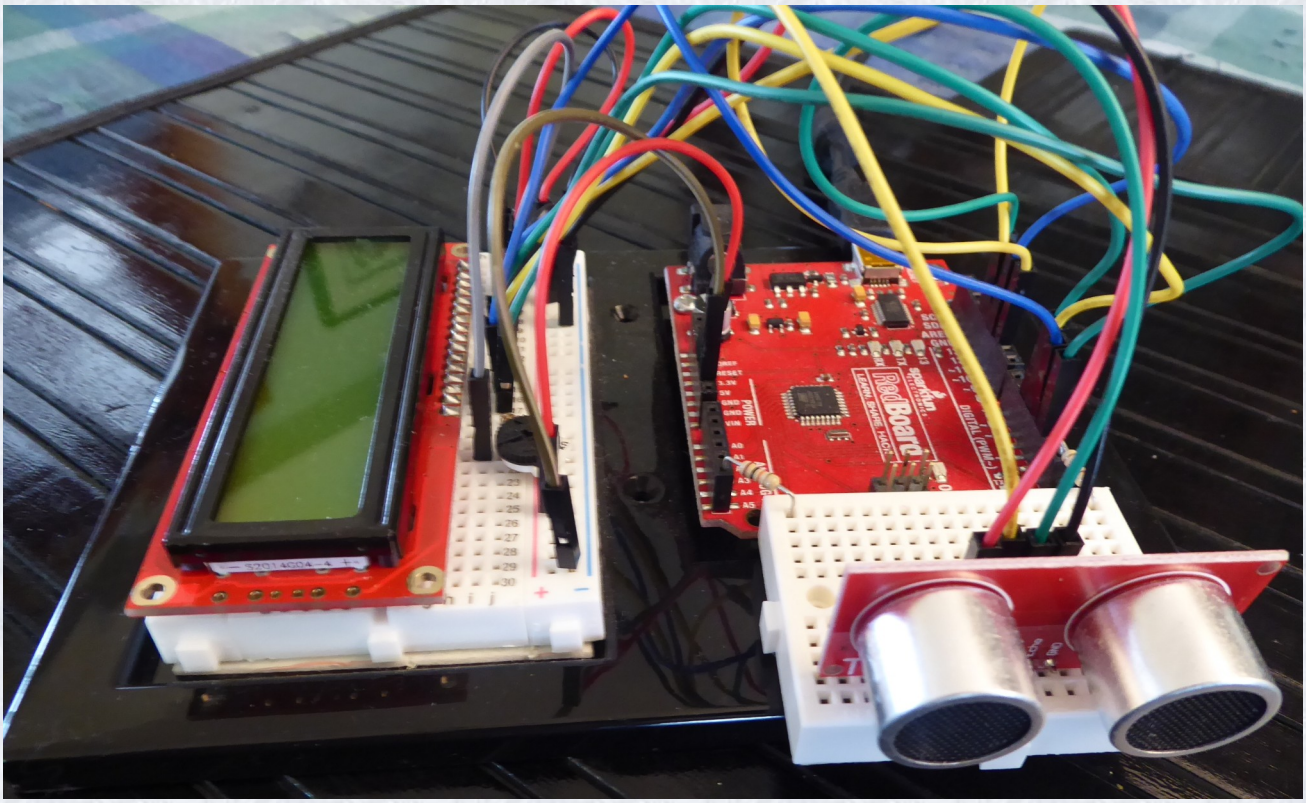
https://components101.com/asset/sites/default/files/component_datasheet/16x2%20LCD%20Datasheet.pdf



Display the distance in centimetres to an object in front of a Proximity Sensor

Sparkfun “Red Board” – Arduino Uno compatible

Sparkfun Proximity Sensor



The code:

2x16 character LCD Display

Sample Code run on an Arduino Uno:

```
/* LCD Pinout
 *  ________________________________________________
 *  |X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  |
 *  |1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16  |
 *  |________________________________________________|
 *  |    HELLO, WORLD                                |
 *  |                                                |
 *  |________________________________________________|
 *
```

```
 *  1 -> GND
 *  2 -> 5V
 *  3 -> Contrast --> 330 ohms resistor
 *  4 -> RS (Register Select)
 *  5 -> RW (Read/Write) --> GND
 *  6 -> EN (Enable)
 *  7 -> D0
 *  8 -> D1
 *  9 -> D2
 *  10 -> D3
 *  11 -> D4
 *  12 -> D5
 *  13 -> D6
 *  14 -> D7
 *  15 -> Backlight Anode --> 2K pot
 *  16 -> Backlight Cathode
 *
 *  LiquidCrystal(RS, EN, D4, D5, D6, D7)
 *
 */

#include <LiquidCrystal.h>
//sensor pins definition
#define trigPin 10
#define echoPin 13
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  float duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  duration = pulseIn(echoPin, HIGH);
  distance = (duration / 2) * 0.0344;

  if (distance >= 400 || distance <= 2){
    lcd.print("Out of range");
    Serial.print("Out of range");
    delay(500);
  }
  else {
    lcd.print(distance);
    lcd.print(" cm");
    Serial.print(distance);
    Serial.print(" cm \n");
    delay(500);
  }
  delay(500);
  lcd.clear();
}
```

Graphical LCD Display

Nokia 3310/5110 type

Driver chip is PCD8544

Pulled from old cell phones

Very inexpensive

84 x 48 pixels

Low power requirements – especially if you don’t need the backlight

Sample code fairly easy to find

Case is fairly robust

LCD Graphics Drawbacks

Not 5V safe – need level shifter or resistors
Used – scratched
Drawing graphics is challenging

Needs lots of CPU memory for anything fancy

example character:
using a 5x8 grid for each character
! (exclamation mark)
0x00, 0x00, 0x5f, 0x00, 0x00

5 in hexidecimal is 0101 in binary
f in hexidecimal is 1111 in binary

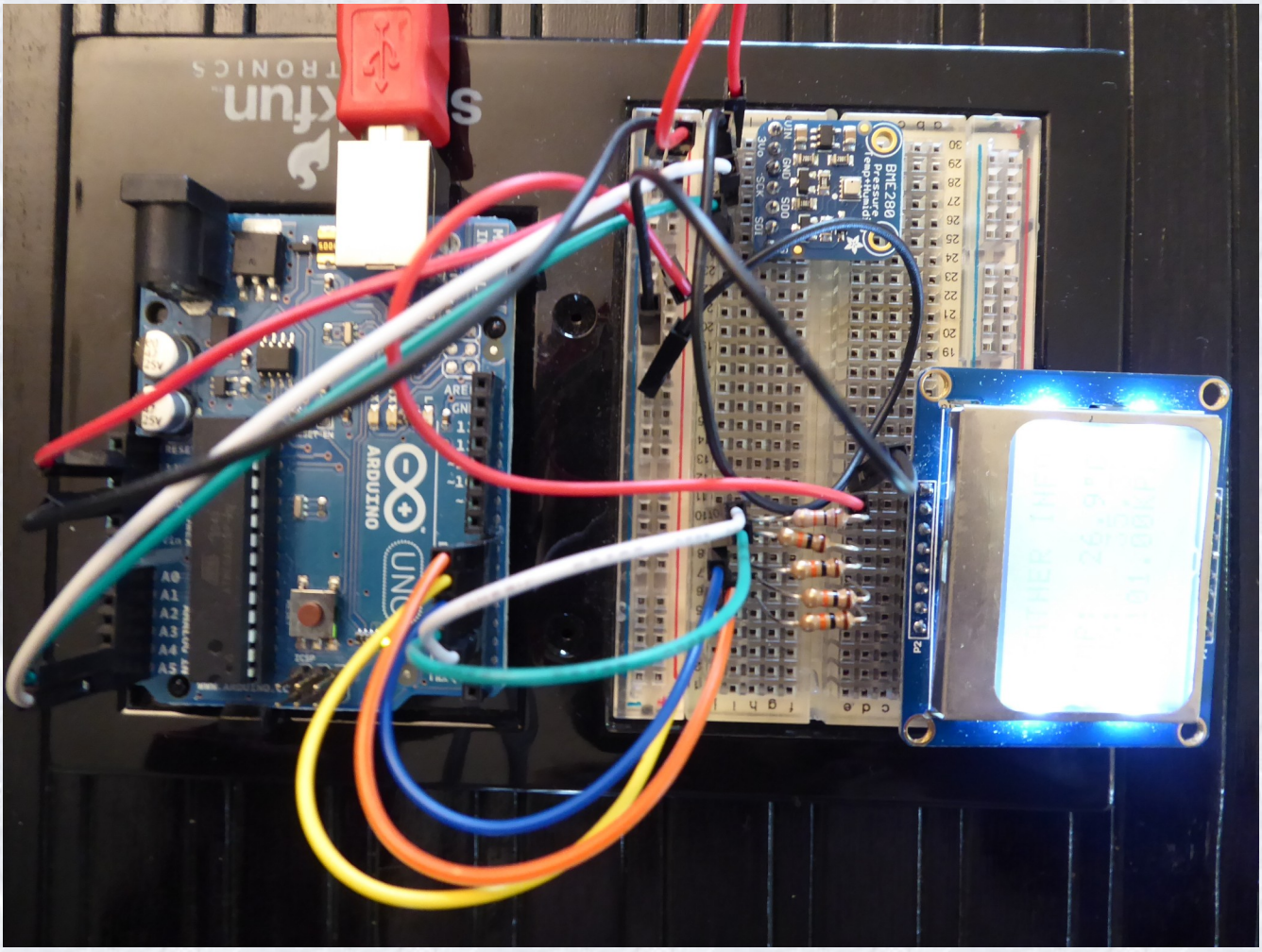
00000000
00000000
01011111
00000000
00000000

home-made “degrees” sign for weather station:

0x00, 0x07, 0x05, 0x07, 0x00

00000000
00000111
00000101
00000111
00000000

Weather Station Project



Nokia 3110 Display

BME 280 Temperature/Humidity/Pressure sensor

Arduino Uno development board

Weather Station Code:

```
/*********************************
 * Hardware connections:
BME280 -> Arduino
GND -> GND
3.3 (VIN) -> 3.3V
SDA (SDI) -> A4
SCL (SCK) -> A5
************************
Nokia 3110 -> Arduino
Pin 1 VCC  -> 3.3V
Pin 2 GND  -> GND
Pin 3 SCE -> 7
Pin 4 RST -> 6
Pin 5 DC -> 5
Pin 6 DOINMOSI -> 4
Pin 7 SCLK -> 3
Pin 8 LED -> n/c
************************
Write to screen:

LCD Initialize
LCD Clear
goto xy
LCD string passes chars to LCD character
LCD character passes bytes to LCD Write
LCD Write goes to write mode, finds the symbol, writes the
bits

************************

 */
```

```
#include <stdint.h>
#include "SparkFunBME280.h”
//Library allows either I2C or SPI, we use I2C for sensor.
#include "Wire.h"
//Display uses SPI
#include "SPI.h"

//Global sensor object
BME280 mySensor;

// Nokia 3110 Pin Definitions:

#define PIN_SCE   7
#define PIN_RESET 6
#define PIN_DC    5
#define PIN_SDIN  4
#define PIN_SCLK  3
//define PIN_VCC   3.3V

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48

static const byte ASCII[][5] =
{
 {0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 "
,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 23 #
,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 24 $
,{0x23, 0x13, 0x08, 0x64, 0x62} // 25 %
,{0x36, 0x49, 0x55, 0x22, 0x50} // 26 &
,{0x00, 0x05, 0x03, 0x00, 0x00} // 27 '
,{0x00, 0x1c, 0x22, 0x41, 0x00} // 28 (
,{0x00, 0x41, 0x22, 0x1c, 0x00} // 29 )
,{0x14, 0x08, 0x3e, 0x08, 0x14} // 2a *
,{0x08, 0x08, 0x3e, 0x08, 0x08} // 2b +
,{0x00, 0x50, 0x30, 0x00, 0x00} // 2c ,
,{0x08, 0x08, 0x08, 0x08, 0x08} // 2d -
,{0x00, 0x60, 0x60, 0x00, 0x00} // 2e .
```

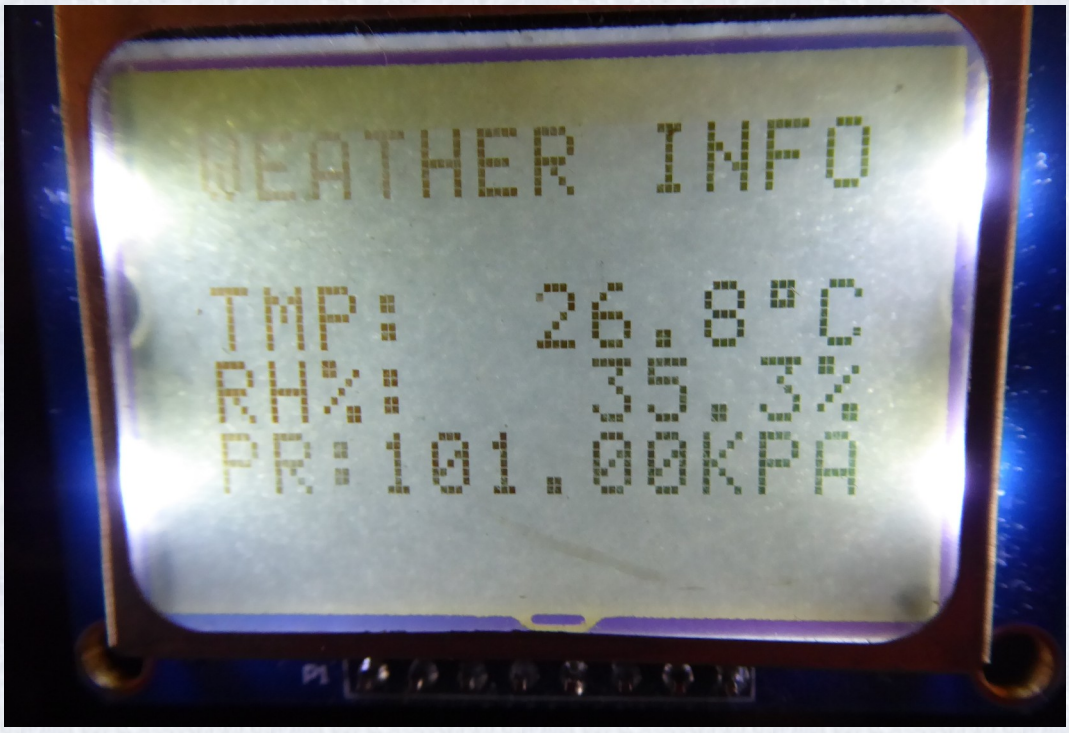
```
,{0x20, 0x10, 0x08, 0x04, 0x02} // 2f /
,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 30 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 31 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 32 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 33 3
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 34 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 35 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 36 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 37 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 38 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 39 9
,{0x00, 0x36, 0x36, 0x00, 0x00} // 3a :
,{0x00, 0x56, 0x36, 0x00, 0x00} // 3b ;
,{0x08, 0x14, 0x22, 0x41, 0x00} // 3c <
,{0x14, 0x14, 0x14, 0x14, 0x14} // 3d =
,{0x00, 0x41, 0x22, 0x14, 0x08} // 3e >
,{0x02, 0x01, 0x51, 0x09, 0x06} // 3f ?
,{0x32, 0x49, 0x79, 0x41, 0x3e} // 40 @
,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 41 A
,{0x7f, 0x49, 0x49, 0x49, 0x36} // 42 B
,{0x3e, 0x41, 0x41, 0x41, 0x22} // 43 C
,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 44 D
,{0x7f, 0x49, 0x49, 0x49, 0x41} // 45 E
,{0x7f, 0x09, 0x09, 0x09, 0x01} // 46 F
,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 47 G
,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 48 H
,{0x00, 0x41, 0x7f, 0x41, 0x00} // 49 I
,{0x20, 0x40, 0x41, 0x3f, 0x01} // 4a J
,{0x7f, 0x08, 0x14, 0x22, 0x41} // 4b K
,{0x7f, 0x40, 0x40, 0x40, 0x40} // 4c L
,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 4d M
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
,{0x7f, 0x09, 0x09, 0x09, 0x06} // 50 P
,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 51 Q
,{0x7f, 0x09, 0x19, 0x29, 0x46} // 52 R
,{0x46, 0x49, 0x49, 0x49, 0x31} // 53 S
,{0x01, 0x01, 0x7f, 0x01, 0x01} // 54 T
,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 55 U
,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 56 V
,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 57 W
,{0x63, 0x14, 0x08, 0x14, 0x63} // 58 X
```

```
,{0x07, 0x08, 0x70, 0x08, 0x07} // 59 Y
,{0x61, 0x51, 0x49, 0x45, 0x43} // 5a Z
,{0x00, 0x7f, 0x41, 0x41, 0x00} // 5b [
,{0x00, 0x07, 0x05, 0x07, 0x00} // 5c \
,{0x00, 0x41, 0x41, 0x7f, 0x00} // 5d ]
,{0x04, 0x02, 0x01, 0x02, 0x04} // 5e ^
,{0x40, 0x40, 0x40, 0x40, 0x40} // 5f _
,{0x00, 0x07, 0x05, 0x07, 0x00} // 60 degrees sign mapped to `
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ←
,{0x78, 0x46, 0x41, 0x46, 0x78} // 7f →
};

void LcdCharacter(char character)
{
  LcdWrite(LCD_D, 0x00);
  for (int index = 0; index < 5; index++)
  {
    LcdWrite(LCD_D, ASCII[character - 0x20][index]);
  }
  LcdWrite(LCD_D, 0x00);
}

void LcdClear(void)
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}


void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
  digitalWrite(PIN_RESET, LOW);
```

```
  digitalWrite(PIN_RESET, HIGH);
  LcdWrite(LCD_C, 0x21 );  // LCD Extended Commands.
  LcdWrite(LCD_C, 0xB1 );  // Set LCD Vop (Contrast).
  LcdWrite(LCD_C, 0x04 );  // Set Temp coefficent. //0x04
  LcdWrite(LCD_C, 0x14 );  // LCD bias mode 1:48. //0x13
  LcdWrite(LCD_C, 0x20 );  // LCD Basic Commands
  LcdWrite(LCD_C, 0x0C );  // LCD in normal mode.

}

void LcdString(char *characters)
{
  while (*characters)
  {
    LcdCharacter(*characters++);
  }
}

void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}
// gotoXY routine to position cursor
// x - range: 0 to 84
// y - range: 0 to 5
void gotoXY(int x, int y) {
  LcdWrite(0, 0x80 | x);  // Column.
  LcdWrite(0, 0x40 | y);  // Row.
}

void setup(void)
{
  LcdInitialise();
  LcdClear();
  gotoXY(0,0);
  LcdString("WEATHER STATION");
  mySensor.settings.commInterface = I2C_MODE;
  mySensor.settings.I2CAddress = 0x77;
  mySensor.settings.runMode = 3; //Normal mode
```

```
  mySensor.settings.tStandby = 0;
  mySensor.settings.filter = 0;
  mySensor.settings.tempOverSample = 1;
  mySensor.settings.humidOverSample = 1;
  mySensor.settings.pressOverSample = 1;
  delay(1000);
  mySensor.begin();

  Serial.begin(57600);
  Serial.print("Program Started\n");
  Serial.print("Starting BME280... result of .begin():
0x");

  //Calling .begin() causes the settings to be loaded
  delay(10);  //Make sure sensor had enough time to turn
on. BME280 requires 2ms to start up.
  Serial.println(mySensor.begin(), HEX);

  Serial.println();

}

void loop(void)
{
  float temp=0;
  char tempStr[15];
  float humid=0;
  char humidStr[15];
  float pressr=0;
  char pressrStr[15];
  LcdClear();
  gotoXY(0,0);
  LcdString("WEATHER INFO");
  gotoXY(0,2);
  temp=mySensor.readTempC();
  dtostrf(temp,4,2,tempStr);
  LcdString("TMP:  ");
  //LcdString((uint8_t)mySensor.readTempC());
  LcdString(tempStr);
  LcdString("\C");
  //delay(1000);
  gotoXY(0,3);
```

```
  humid=mySensor.readFloatHumidity();
  dtostrf(humid,3,1,humidStr);
  LcdString("RH%:   ");
  LcdString(humidStr);
  LcdString("%");
  //delay(1000);
  gotoXY(0,4);
  pressr=mySensor.readFloatPressure();
  pressr=pressr/1000;
  dtostrf(pressr,4,2,pressrStr);
  LcdString("PR:");
  LcdString(pressrStr);
  LcdString("KPA");
  //delay(1000);
  Serial.print("Temperature: ");
  Serial.print(mySensor.readTempC(), 2);
  Serial.println(" C");
  Serial.print("temp = ");
  Serial.println(temp);
  Serial.print("pressure = ");
  Serial.println(pressr);
  Serial.print("humidity = ");
  Serial.println(humid);
  delay(10);
}
```



Colour TFT Displays!!!

Bright, colourful, inexpensive

Power efficient

Easy to see in low light

TFT – thin film transistor, limited viewing angle

IPS – In-Plane switching, better colour, brighter, better viewing angle

BUT

more complicated to code, even with libraries

Weather Station Project 2.0

I used CircuitPython and Mu

Use Adafruit's displayio library

Microcontroller is a SAMD51

Same BME 280 as before

Display is 240x320 TFT IPS with ST7789 controller/driver,

2" diagonal screen 18 bit colour

The Plan

Import the driver

Define the display bus

Define the display resolution, starting point

Define font file

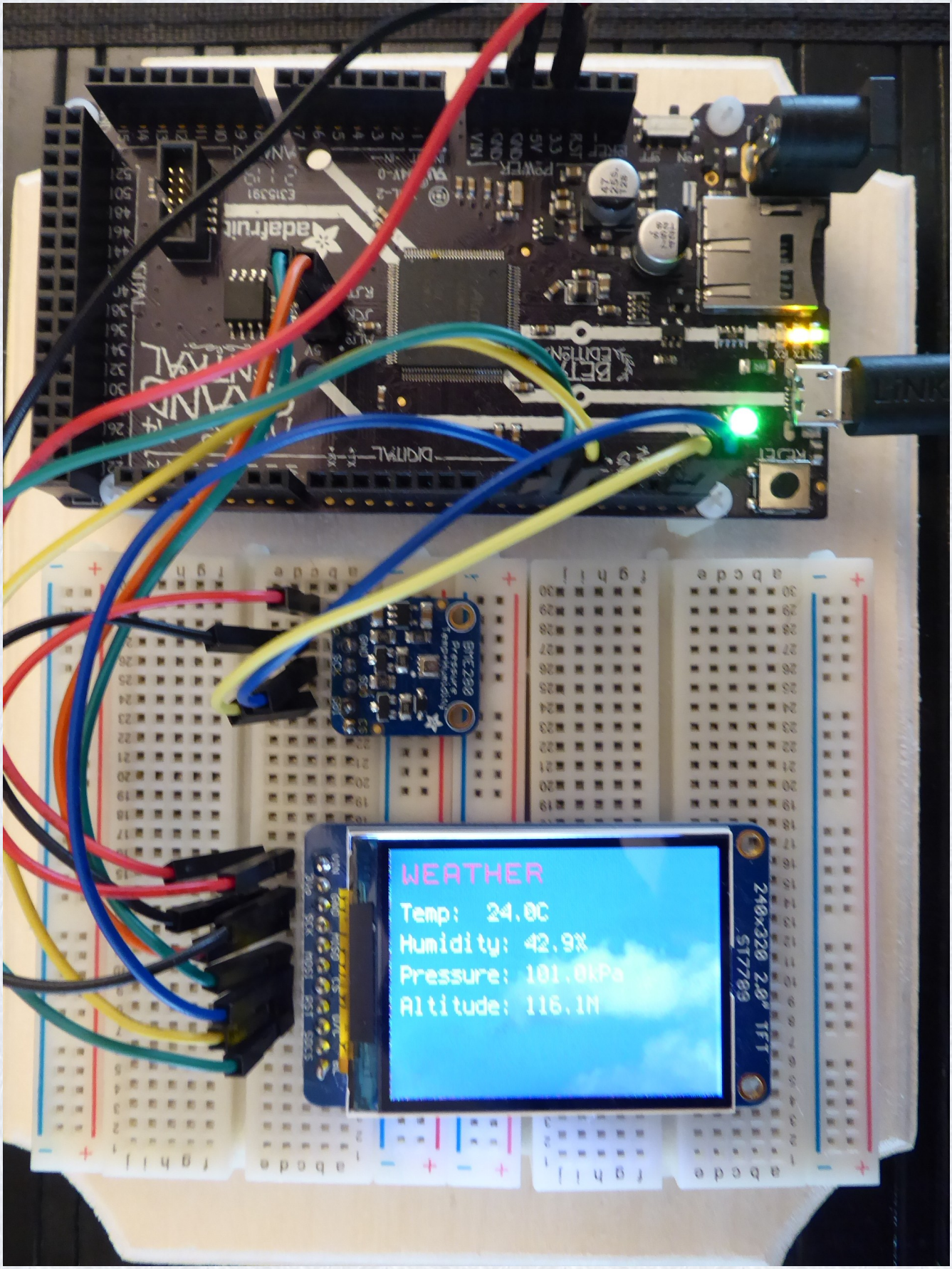
Define the display group

Define a tile with a bitmap and colour palette

Append the tile to the display group

To print text

- Load the font file
- Set up text label, define with font, variable name, colour
- Define x and y starting location
- Append the label to the display


WEATHER
Temp: 24.0C
Humidity: 42.9%
Pressure: 101.0kPa
Altitude: 116.1M
240x320 2.8" TFT
ST7789

Weather Station 2.0 Code:

```
#Circuitpython code for Metro and Grand Central
# ST7789 2" TFT Display
# BME 280 Temperature/Pressure/Humidity sensor
"""
This test will initialize the Adafruit ST7789 240x320
display using displayio
and display weather information on a teal background
with "WEATHER" in red

WIRING
------
ST7789      Grand Central
Vin         3V
3.3V        n/c
GND         G
SCK         SCK
MISO        n/c
MOSI        MOSI
TFT_CS      D9
RST         D11
D/C         D10
CARD_CS     n/c
LITE        n/c

BME280(i2c) Grand Central
VIN         3V
GND         GND
SCK         SCL
SDI         SDA

"""
import board
import displayio
import terminalio
import time
import busio
import adafruit_bme280
import adafruit_imageload
from adafruit_bitmap_font import bitmap_font
```

```
from adafruit_st7789 import ST7789
from adafruit_display_text.label import Label

spi = board.SPI()
#chip select
tft_cs = board.D9
#data/command
tft_dc = board.D10
HEAD_STRING = "WEATHER"
displayio.release_displays()
display_bus = displayio.FourWire(spi, command=tft_dc,
chip_select=tft_cs, reset=board.D11)

display = ST7789(display_bus, width=320, height=240,
colstart=1, rowstart=1, rotation=90)
# load the heading font
large_font_name = "/fonts/Gomme10x20n.bdf"
large_font = bitmap_font.load_font(large_font_name)
large_font.load_glyphs(HEAD_STRING.encode('utf-8'))
# Make the display context - Display group - collection of
tile grids
my_display = displayio.Group(
    max_size=20,
    scale=2)
display.show(my_display)

bitmap, palette =
adafruit_imageload.load("/weather_paint.bmp",
    bitmap=displayio.Bitmap,
    palette=displayio.Palette)
my_background = displayio.TileGrid(bitmap,
pixel_shader=palette)

my_display.append(my_background)
# Weather sensor setup
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# change this to match the location's pressure (hPa) at sea
level
bme280.sea_level_pressure = 1024.00
# set up strings, 2 decimal places
```

```
TEMP_STRING = "Temp:  " + str(round(bme280.temperature, 2))
+ "C"
HUMID_STRING = "Humidity: " + str(round(bme280.humidity,
2)) + "%"
PRESS_STRING = "Pressure: " +
str(round((bme280.pressure/10), 2)) + "kPa"
ALT_STRING = "Altitude: " + str(round(bme280.altitude, 2))
+ "M"
# set up the text in the screen
BACKGROUND_TEXT_COLOR = 0xFFFFFF
HEADING_BACKGROUND = 0xFF0000
font = terminalio.FONT
# text gets placed as labels.  Labels need a size and a
location.
heading_label = Label(large_font, text=HEAD_STRING,
color=0x00FF00)
(x, y, w, h) = heading_label.bounding_box
heading_label.x = 5
heading_label.y = 10
heading_label.color = HEADING_BACKGROUND
my_display.append(heading_label)
temp_label = Label(font, text=TEMP_STRING, color=0x00FF00)
(x, y, w, h) = temp_label.bounding_box
temp_label.x = 5
temp_label.y = 30
temp_label.color = BACKGROUND_TEXT_COLOR
my_display.append(temp_label)
humid_label = Label(font, text=HUMID_STRING,
color=0x00FF00)
(x, y, w, h) = humid_label.bounding_box
humid_label.x = 5
humid_label.y = 45
humid_label.color = BACKGROUND_TEXT_COLOR
my_display.append(humid_label)
pressure_label = Label(font, text=PRESS_STRING,
color=0x00FF00)
(x, y, w, h) = pressure_label.bounding_box
pressure_label.x = 5
pressure_label.y = 60
pressure_label.color = BACKGROUND_TEXT_COLOR
my_display.append(pressure_label)
```

```
altitude_label = Label(font, text=ALT_STRING,
color=0x00FF00)
(x, y, w, h) = altitude_label.bounding_box
altitude_label.x = 5
altitude_label.y = 75
altitude_label.color = BACKGROUND_TEXT_COLOR
my_display.append(altitude_label)
print("Hello, Metro")
while True:
    TEMP_STRING = "Temp:  " + str(round(bme280.temperature,
1)) + "C"
    HUMID_STRING = "Humidity: " +
str(round(bme280.humidity, 1)) + "%"
    PRESS_STRING = "Pressure: " +
str(round((bme280.pressure/10), 1)) + "kPa"
    ALT_STRING = "Altitude: " + str(round(bme280.altitude,
1)) + "M"
    temp_label.text = TEMP_STRING
    humid_label.text = HUMID_STRING
    pressure_label.text = PRESS_STRING
    altitude_label.text = ALT_STRING
    time.sleep(5)
    pass
```