Arduino Nano: Adding a screen

ECU Makerspace Group

Written by Julia Szymanski

Getting started…

In this workshop, we are going to **add a new gadget: a screen.**

We'll connect the screen to our Arduino and display a printed message first, before we later combine it with the ultrasonic ranger and display the measured distance to the screen.

Adding a screen

You may notice the different abbreviations on the left side next to the screen.

RST – Reset Display

CE – Chip Enable

DC – Data Command

DIN – Serial Data pin

CLK- Serial Clock Pin

VCC – power supply

BL – Backlight

GND - Ground

When connecting the screen to the Arduino, **it is important that we connect it correctly**, as per the following:

NOKIA DISPLAY	ARDUINO
Pin1 (RST)	D3
Pin2 (CE)	D4
Pin3 (DC)	D5
Pin4 (DIN)	D6
Pin5 (CLK)	D7
Pin6 (VCC)	3.3V
Pin7 (BL)	3.3V
Pin8 (GND)	GND

Arduino Nano: Adding a screen

ECU Makerspace Group

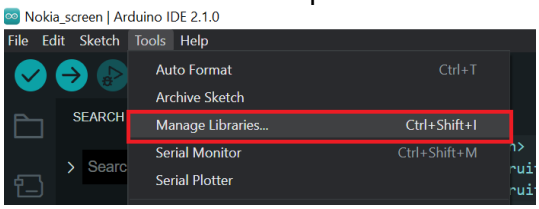
Written by Julia Szymanski

To make sure that we can use the screen, there is a few extra functionalities (also referred to as **libraries**) that we need to include or install.

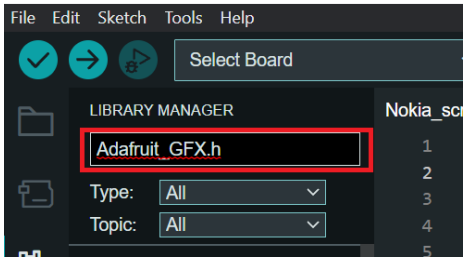
First, at the beginning of the file, we include `#include <SPI.h>`. As this is already included with every Arduino platform, we only need to include it, but there is no need to install it separately.

We also need to include `#include <Adafruit_GFX.h` which is a **graphics library** that is required for the screen. This one has to be **installed manually** – just follow the below steps:

1. Click on **Tools** at the top of the IDE and select **Manage Libraries**



2. In the **Search**, enter **Adafruit_GFX.h**



3. Install **Adafruit GFX Library** by Adafruit. Make sure you install version 1.11.5.

Arduino Nano: Adding a screen

ECU Makerspace Group

Written by Julia Szymanski

There is **another library** that we need to install and include - `#include <Adafruit_PCD8544.h>` , which is the library specially for the screen that we are using.

This library also needs to be installed. In order to do this, follow the same steps as before, but instead enter **Adafruit_PCD8544.h** in the search. Make sure you install the latest version of it and include the #include statement at the beginning of your code.

Before we can display a message to the screen, we need to make sure that the program knows which pins the screen is connected to. For this, we set a variable called **Adafruit_PCD8544** and set it to the pins 7,6,5,4,3 that are connected to the screen (which is here represented by Adafruit_PCD8544.

```
Adafruit_PCD8544 display = Adafruit_PCD8544(7,6,5,4,3);
```

Your code should now look like this:

Nokia_screen.ino

```
1  #include <SPI.h>
2  #include <Adafruit_GFX.h>
3  #include <Adafruit_PCD8544.h>
4
5  Adafruit_PCD8544 display = Adafruit_PCD8544(7,6,5,4,3);
6
```

Now we are ready to display a message to the screen! Well done so far! 😊

Display a message on the screen

Before we can begin to write any code to show a message on the screen, we need to set up the data transfer rate again in the **void setup()** function. This time we need to set it to

```
Serial.begin(115200);
```

instead of 9600, as the screen requires a higher data transfer rate.

Next, we need to give the program instructions to start setting up the display by typing

```
display.begin();
```

just below serial.begin(115200);

Using the prefix display refers to anything related to the screen that we have connected.

It is also important that we set contrast on the screen, so we can actually see the message. We set the contrast to 50 here. You can try to change this up if you prefer a different contrast

```
display.setContrast(50);
```

As we might change the message on the screen, we need to clear the display of any other text before we display a new one, otherwise the text may overlap

```
display.clearDisplay();
```

Depending on how much text we want to display, we need to set a text size so the entire text fits on the screen. For now, we are going to set the text size to 1, but you can change this later on if you like.

```
display.setTextSize(1);
```

There is also the option to set it to a different text colour – for now we set it to BLACK.

```
display.setTextColor(BLACK);
```

To show text on the screen, we use **display.println("TEXT")** (l is a lowercase L)

```
display.println("Hello, World!");
```

You can change up the text from "Hello, World!" to something more creative, if you like.

The last step is to tell the program to display all of the above on the display screen.

```
display.display();
```

All of this is within the void setup() function and the void loop() function stays empty for now.

Your code should now look like this:

Nokia_screen.ino

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

Adafruit_PCD8544 display = Adafruit_PCD8544(7,6,5,4,3);

void setup()   {
  Serial.begin(115200);
  display.begin();
  display.setContrast(50);

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(BLACK);
  display.println("Hello, World!");
  display.display();
}

void loop() {

}
```

Arduino Nano: Adding a screen

ECU Makerspace Group

Written by Julia Szymanski

Show the distance from the ultrasonic ranger on the screen

Now it is time to **show the distance from the ultrasonic ranger on the screen**.

First, we are going to connect our ultrasonic ranger to the screen as we did in Session 2. If you are unsure about this, you may wish to refer to the instructions from Session 2. We also need to make sure the screen is connected as per instructions on the first pages of this workshop.

First we need to make sure to **include the libraries** required for the screen and the ultrasonic ranger, which are:

```
#include <Ultrasonic.h>
```

```
#include <Ultrasonic.h>
```

```
#include <SPI.h>
```

```
#include <Adafruit_GFX.h>
```

```
#include <Adafruit_PCD8544.h>
```

We also need to **define the Rangerpin** connected to digital pin 11 and set up the display connected to digital pins 13, 12, 11, 10 and 9.

```
#define RANGERPIN 11
Ultrasonic ultrasonic(11);

Adafruit_PCD8544 display = Adafruit_PCD8544(13,12,11,10,9);
```

Once we have done that, we can move to the void set up() function.

Since we are using the display that requires a higher data transfer rate, we are going to set it to 115200 again

```
Serial.begin(115200);
```

Next, we tell our program that the following instructions are related to the screen by adding

```
  display.begin();
```

After that, we **clear the display** of any other text that may be still on there, **set the contrast, text size and text colour** as we have done in the first part of this workshop

This is what your code should now look like:

Session3.ino

```
#include <Ultrasonic.h>
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
//#include <Wire.h>

#define RANGERPIN 11
Ultrasonic ultrasonic(11);

Adafruit_PCD8544 display = Adafruit_PCD8544(13,12,11,10,9);


void setup()
{
  Serial.begin(115200);
  display.begin();
  display.clearDisplay();
  display.setContrast(50);
  display.setTextSize(1);
  display.setTextColor(BLACK);

}
```

In the last step, we move on to the void loop() function so we can **print the distance** from the ultrasonic ranger to the screen.

First we need to **create a variable** that holds the value of the measured distance. We are going to name it RangeInCentimeters and assign it to ultrasonic.MeasureInCentimeters() which is how we can retrieve the data from the ultrasonic ranger.

```
long RangeInCentimeters;
   RangeInCentimeters = ultrasonic.MeasureInCentimeters();
```

Before we print out the actual value, we just add a simple print statement that prints “Distance in Centimeters” so that we can see on our screen what we are printing out

```
  display.println("Distance in centimeters:");
```

Since we have the value of the measured distance assigned to a variable already, we can just proceed by printing out the value of that variable

```
display.print(RangeInCentimeters);
```

Lastly, we add another print statement that prints “cm” at the end, so that we know that the value is in centimeters. We also add a **delay of 2000**, so that we have a chance to read the value on the screen before the next one appears.

```
display.println("cm");
  delay(2000);
```