# NFC Spotify Player using Arduino

## Pre-build

Before beginning assembling the Spotify Player, I am testing the two displays I am deliberating in using, a 2.4” TFT LCD and a 0.96” LCD display. This is mostly to get comfortable with using a breadboard, wires, Arduinos and external modules.

For the project (maybe only for the pre-build, depending on how it goes) I will be using only ChatGPT as my assistant and guide to test its usefulness in such a setting.

### 2.4” TFT LCD

After some wiring, I have managed to connect the display to my Arduino, however I am still struggling to display things that aren’t just the standard white background. Chat GPT has been helpful, mainly in identifying the resistor I needed to use for the LED pin, but it has also been inconsistent in what pins to connect and to where, meaning I have had to rewire some connections.

The following day, I tried wire the 2.4” display again, using 3k3 and 2k2 resistors on the data pins to lower the voltage level to work at the 3.3v logic that the display used. However, after a lot of different configurations, I could still not get it to display anything other than a blank white screen.

Since I was not planning to use this display in the final project anyway, I have decided to move onto the 0.96” display.

I have also abandoned the notion of only using Chat GPT as it was not very useful and often contradicted itself. I did find that using other resources such as Google or Youtube initially and then using Chat GPT to refine any points I was unsure about was quite effective.

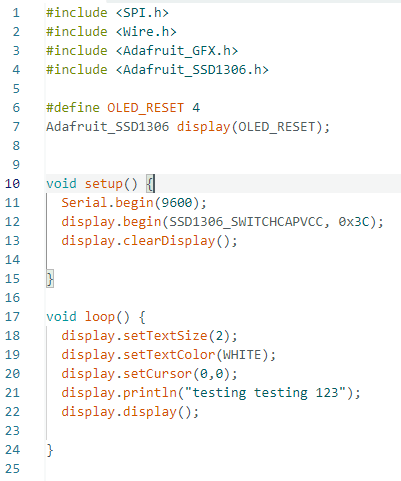
### 0.96” OLED

This display uses the I2C protocol, meaning that it only has 4 connections and should be a lot easier to work with, and it should leave plenty of digital pins for the other components in the project. However, this also means that its display capabilities are limited to just one colour.

I wired it up with the following configuration. The colour of the background is the colour of wire used.

|  |  |
| --- | --- |
| Arduino | OLED Display |
| 5V | VCC |
| GND | GND |
| A4 | SCL |
| A5 | SDA |

I managed to display some test text on the screen with the following code.



PIC OF TESTING DISPLAY HERE

After getting the display to work, I also decided to carry on testing some of the other components, first of which is the KC11B04 keypad.

### 4 Button Keypad

The keypad is very simple and only has 3 connections.

|  |  |
| --- | --- |
| Arduino | KC11B04 |
| A0 | AD |
| 5V | VCC |
| GND | GND |

The 4 buttons are detected by the output of the AD pin which is a value between 0-1024.

Button Values:

K1: 400

K2: 600

K3: 800

K4: 1024

The labelled keys printed their respective values in the Serial, and using if statements to check if the values are in a range allowed me to detect which buttons are pressed.

I put the following line in the header:

const int AD\_PIN = A0; /\*Keypad pin\*/

And the following lines elsewhere:

void loop() {

  display.setTextSize(2);

  display.setTextColor(WHITE);

  display.setCursor(0,0);

  display.println("testing testing 123");

  display.display();

  int adValue = analogRead(AD\_PIN);

  Serial.println(buttonFromValue(adValue));

}

byte buttonFromValue(int adValue) {

  if (adValue > 300 && adValue < 500){

    return 1;

  }

  if (adValue > 500 && adValue < 700){

    return 2;

  }

  if (adValue > 700 && adValue < 900){

    return 3;

  }

  if (adValue > 900){

    return 4;

  }

  return 0;

}

I noticed that when you touched the back of the keypad, your finger caused interference with the values returned, but the threshold for each keypad button is big enough to not be affected by this when a key is pressed.

Next, I will try to display the number of the button pressed on the OLED display as it is clicked.

void loop() {

  display.setTextSize(3);

  display.setTextColor(WHITE);

  display.setCursor(0,0);

  int adValue = analogRead(AD\_PIN);

  delay(100);

  int button = buttonFromValue(adValue);

  Serial.println(button);

  if (button != lastButton && button != 0){

    lastButton = button;

    display.clearDisplay();

    display.println(lastButton);

    display.display();

  }

}

This is the code I used to implement this. If a new button is pressed, it changes the display to said button. The delay of 100ms is necessary since sometimes the value of the AD pin goes down as you let go of a button, and drops into the undesired threshold. For instance, after pressing 3 and letting go, it sometimes drops into 2 and displays that instead.

PIC OF NUMBERED DISPLAY HERE

## Prototype 1

Now that I am comfortable with the display and keypad, I am going to create my first prototype that connects to my Spotify API and displays the name of the song playing, the artist, and lets me play and skip songs.

**Prototype 1 Components:**

* KC11B04 Keypad
* 0.96” OLED I2C Display
* Arduino Uno
* Jumper wires

**Prototype 1 Aims:**

* Connect to Spotify Account
* Display song name
* Display artist
* Play/pause using keypad
* Go back/skip songs using keypad

First, I tried to create a ‘thing’ on the Arduino IoT cloud but ran into my first problem – the Arduino Uno R3 cannot connect to the internet, so I will have to connect the ESP01 Wifi Module to it before I can proceed.

|  |  |
| --- | --- |
| Arduino | ESP01 |
| 3.3v | 3.3v |
| 3.3v | EN |
| GND | GND |
| D2 | TX |
| D3 | RX |

**NEXT FIND IP OF ESP01 AND THEN USE THAT TO CONNECT TO SPOTIFY API**