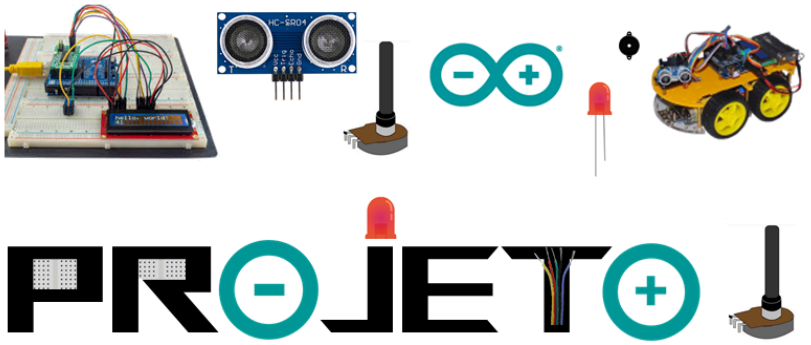

Licenciatura em Engenharia Eletrotécnica e de Computadores

Laboratorial assignment # 6

Displays: TFT screen and 16x2 LCD

Components list:

• Arduino UNO • 1 USB cable • 2 white breadboards • 2 potentiometers	• 1 TFT-LCD display • 1 LCD 16x2 display • 1 220Ω resistor • Software: Arduino IDE

TFT-LCD display

A TFT-LCD display consists of a screen where you can draw text, images and shapes using the TFT library. Its size is 160x128 pixels and it is designed to be used with the Arduino's breadboard by connecting the pinout displayed in the next image.

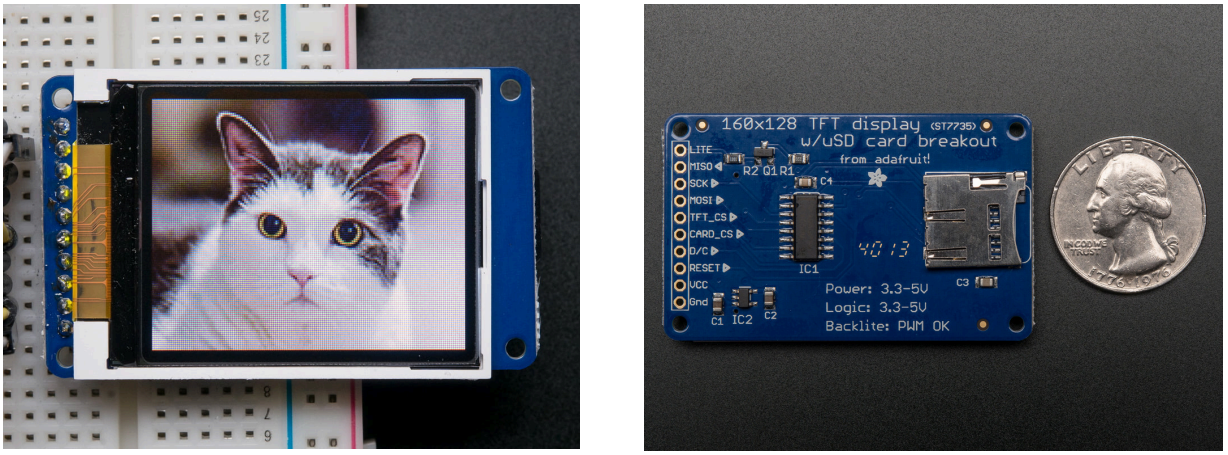

Figure 1. TFT-LCD display module.

1. Circuit and basic code

Mount the LCD display and circuit as presented in Figure 2.

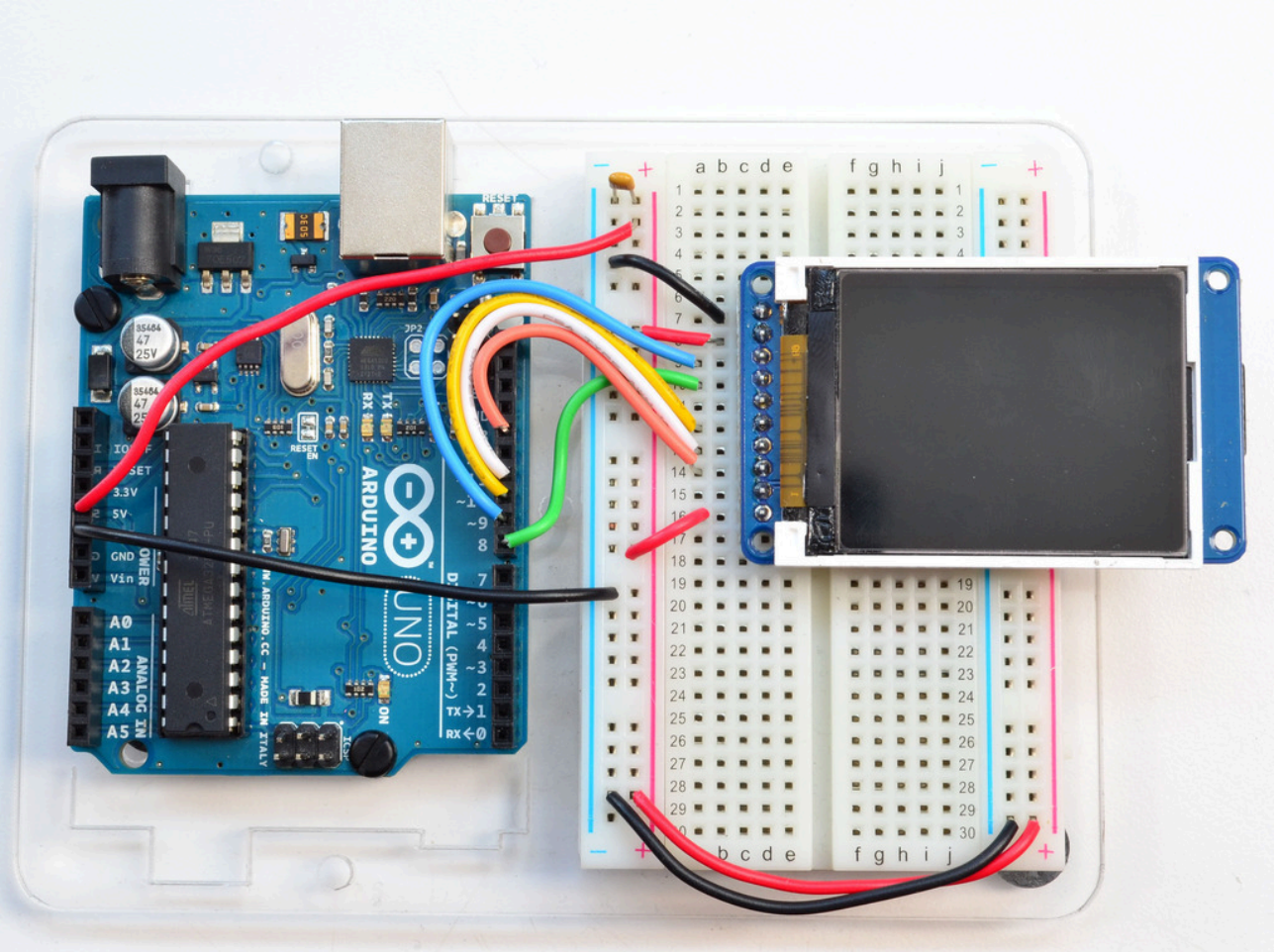

Figure 2. Circuit with a TFT-LCD display module.

- **3-5V Vin** connects to the Arduino **5V** pin - red wires
- **GND** connects to Arduino ground - black wires
- **CLK** connects to SPI clock. On Arduino Uno/Duemilanove/328-based, thats **Digital 13**. On Mega's, its **Digital 52** and on Leonardo/Due its **ICSP-3** (See SPI Connections for more details) - this is the orange wire
- **MOSI** connects to SPI MOSI. On Arduino Uno/Duemilanove/328-based, thats **Digital 11**. On Mega's, its **Digital 51** and on Leonardo/Due its **ICSP-4** (See SPI Connections for more details) - this is the white wire
- **CS** connects to our SPI Chip Select pin. We'll be using **Digital 10** but you can later change this to any pin - this is the yellow wire
- **RST** connects to our TFT reset pin. We'll be using **Digital 9** but you can later change this pin too - this is the blue wire
- **D/C** connects to our SPI data/command select pin. We'll be using **Digital 8** but you can later change this pin too - this is the green wire

Copy this code snippet to your program and test it to make sure the TFT screen is working and the connections are correct.

```
#include <TFT.h>  // Arduino LCD library
#include <SPI.h>

// pin definition for the Uno
#define cs   10
#define dc   8
#define rst  9

TFT TFTscreen = TFT(cs, dc, rst);

// position of the line on screen
int xPos = 0;

void setup() {
  // initialize the serial port
  Serial.begin(9600);

  // initialize the display
  TFTscreen.begin();

  // clear the screen with a pretty color
  TFTscreen.background(250, 16, 200);
  ...
}
```

Since Tinkercad does not include a TFT screen, it is not possible to test circuits with this component in the tool. However, you can use the simulator at wokwi.com to test these circuits. The tool does not include a 160x128 pixels display, as the one provided in classes, but a 320x240 screen instead, so the code presented here will only use part of the screen. However, you can use this to prepare the assignment and test your code.

2. Drawing shapes

In this exercise, you will draw some shapes on the TFT screen. The program should first draw a square, with any background color and stroke color you choose. Then, the program should change the background color and draw a triangle (with a different stroke color). Finally, the program should draw a rectangle. The program should then loop back to the first state. Each shape should be kept for 2 seconds. Every time the shape changes, the background and stroke color should also change.

To achieve this objective, you can use the following functions:

- `void background (red, green, blue)` – fills the entire screen with a predefined color
- `int height ()` – returns the number of rows of the screen (128 rows in normal mode)
- `void line (x1, y1, x2, y2)` – draws a line connecting the points (x1, y1) and (x2, y2)
- `int map (value, min, max, Min, Max)` – interpolates a value in the interval [min, max] to fit in the interval [Min, Max]. Returns the new value
- `void stroke (red, green, blue)` – selects the color to be used in the display

Notice that to call these functions from the TFT library, you must call them using an object of the class TFT. The code snippet of the previous point defines the `TFTscreen` object, so that you can call functions as `TFTscreen.background()`, for instance.

3. Printing text

Change the previous example to include the name of each shape (or some abbreviation of it) inside it. You may find the following functions useful:

- `void setTextSize(int)` – sets the size of the font to print to the screen
- `void text(string, xpos, ypos)` – prints the text in string at position (x, y)
- `void String.toCharArray(charArray, size)` – converts a string to a `char` array.

Please notice that you may have to change the position to draw the graph (previous point).

4. Drawing a line graph

In this exercise, you will draw a graph that indicates the value read from a potentiometer during runtime. The graph should change in real-time as we move the cursor of the potentiometer. The cursor should move from left to right, updating the height value (up and down) according to the value read from the potentiometer (plotting a continuous line).

- `void background (red, green, blue)` – fills the entire screen with a predefined color
- `int height ()` – returns the number of rows of the screen (128 rows in normal mode)
- `void line (x1, y1, x2, y2)` – draws a line connecting the points (x1, y1) and (x2, y2)
- `int map (value, min, max, Min, Max)` – interpolates a value in the interval [min, max] to fit in the interval [Min, Max]. Returns the new value
- `void stroke (red, green, blue)` – selects the color to be used in the display

The following code structure works as the basis of the solution to draw a graph from the reading of a potentiometer's value.

```
void loop() {
  // read the sensor and map it to the screen height
  ...

  // print out the height to the serial monitor
  ...

  // draw a line in a nice color
  ...

  // if the graph has reached the screen edge
  // erase the screen and start again
  ...


  //wait a bit
  delay(50);
}
```

16x2 LCD display

Let's now program a different kind of display. An LCD display consists of a screen where you can draw text and shapes using the <LiquidCrystal.h> library. You can control 16x2 characters and it is designed to be used with the Arduino's breadboard by connecting the pinout displayed in the next image.

You can prepare and test circuits with 16x2 LCD displays in Tinkercad.

Pin Description and Wiring Diagram

Pin No.	Symbol	External Connection	Function Description
1	V_{SS}	Power Supply	Ground
2	V_{DD}	Power Supply	Supply Voltage for logic (+5.0V)
3	V_0	Adj Power Supply	Power supply for contrast (approx. 0.6V)
4	RS	MPU	Register select signal. RS=0: Command, RS=1: Data
5	R/W	MPU	Read/Write select signal, R/W=1: Read R/W: =0: Write
6	E	MPU	Operation enable signal. Falling edge triggered.
7-10	DB0 – DB3	MPU	Four low order bi-directional three-state data bus lines. These four are not used during 4-bit operation.
11-14	DB4 – DB7	MPU	Four high order bi-directional three-state data bus lines.
15	NC	-	No Connect
16	NC	-	No Connect

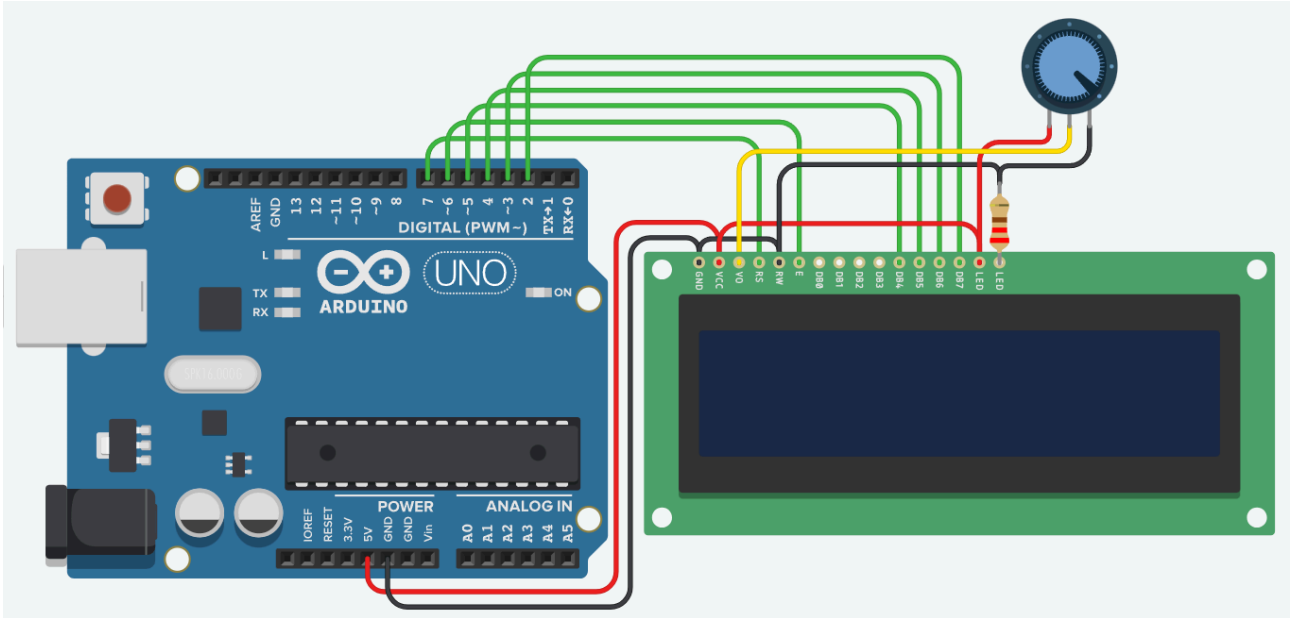

Figure 3. LCD 16x2 display module connections.

5. Circuit and basic code

The 16x2 LCD screen can be controlled using the LiquidCrystal library. Next, you can find some of the functions that you can use to manipulate the screen:

- **`scrollDisplayRight()`** – Scrolls the contents of the display (text and cursor) one space to the right.
- **`scrollDisplayLeft()`** – Scrolls the contents of the display (text and cursor) one space to the left.
- **`blink()`** – Turns on the blinking cursor.
- **`noBlink()`** – Turns off the blinking cursor.
- **`clear()`** – Clears the LCD screen and positions the cursor in the upper-left corner.
- **`setCursor()`** – Sets the cursor location at which subsequent text written to the LCD will be displayed.
- **`noCursor()`** – Hides the LCD cursor.

Mount the LCD display and circuit as presented in Figure 3 and test the code snippet shown below.

Note: DO NOT remove the circuit from the previous exercises (with the TFT screen)! You will need it again. Instead, connect the 16x2 LCD using the remaining digital pins. You may use an extra breadboard for the following exercises.

```
#include<LiquidCrystal.h>
// Create an LCD object and connect control wires
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
unsigned long lastUpdate;


void setup(){
// Specify the LCD's number of columns and rows.
  lcd.begin(16, 2);
  lcd.setCursor(3,0);
  lcd.print("Projeto 1!");
  lcd.setCursor(0,1);
  lcd.print("LCD 16x2 Screen");
  lastUpdate = millis();
}

void loop(){
  if((millis() - lastUpdate) > 250) {
    lcd.scrollDisplayRight();
    lastUpdate = millis();
  }
}
```

6. Custom characters

In addition to printing text, you can create custom characters and symbols to print on the LCD (for example, to show symbols or create graphics). The following code shows how to design your own characters and use them. Test the following code and adapt it to print each of the 8 custom

characters in a random location on the screen. After 5 seconds, the screen should be cleared and the process repeated (all custom characters printed to new random locations).

You can also design your own custom symbols to replace those given - creativity is encouraged!

```
#include<LiquidCrystal.h>
// Create an LCD object and connect control wires
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
unsigned long lastUpdate;

// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};
byte Bell[] = {
  B00100,
  B01110,
  B01110,
  B01110,
  B11111,
  B00000,
  B00100,
  B00000
};
byte Alien[] = {
  B11111,
  B10101,
  B11111,
  B11111,
  B01110,
  B01010,
  B11011,
  B00000
};
byte Check[] = {
  B00000,
  B00001,
  B00011,
  B10110,
  B11100,
  B01000,
  B00000,
  B00000
};
byte Speaker[] = {
  B00001,
  B00011,
  B01111,
  B01111,
  B01111,
  B00011,
  B00001,
  B00000
};
byte Sound[] = {
  B00001,
  B00011,
  B00101,
```

```
  B01001,
  B01001,
  B01011,
  B11011,
  B11000
};
byte Skull[] = {
  B00000,
  B01110,
  B10101,
  B11011,
  B01110,
  B01110,
  B00000,
  B00000
};
byte Lock[] = {
  B01110,
  B10001,
  B10001,
  B11111,
  B11011,
  B11011,
  B11111,
  B00000
};


void setup() {
  // Specify the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Create a new characters:
  lcd.createChar(0, Heart);
  lcd.createChar(1, Bell);
  lcd.createChar(2, Alien);
  lcd.createChar(3, Check);
  lcd.createChar(4, Speaker);
  lcd.createChar(5, Sound);
  lcd.createChar(6, Skull);
  lcd.createChar(7, Lock);
  // Clears the LCD screen:
  lcd.clear();
}


void loop(){
  lcd.setCursor(0, 1);
  lcd.write(byte(0));
  lcd.setCursor(2, 1);
  lcd.write(byte(1));
  lcd.setCursor(4, 1);
  lcd.write(byte(2));
  lcd.setCursor(6, 1);
  lcd.write(byte(3));
  lcd.setCursor(8, 1);
  lcd.write(byte(4));
  lcd.setCursor(10, 1);
  lcd.write(byte(5));
  lcd.setCursor(12, 1);
  lcd.write(byte(6));
  lcd.setCursor(14, 1);
  lcd.write(byte(7));
}
```

7. Dynamic Bar Graph

Adapt the code you developed for exercise 4 to use a TFT screen and a 16x2 LCD display simultaneously. The TFT should keep the same functionality as before - drawing a line plot depending on the value read from the potentiometer. The 16x2 LCD should print the value read from the analog input (potentiometer) on the lower line and a bar graph proportional to the value read in the upper line (growing from the left to the right). If the analog input reads the maximum value, the bar should fill the screen.

Use a custom character (fully filled square) to draw the bar graph, as shown on the left in Figure 4.

Optional objective for extra points: You can use multiple custom characters to increase the precision of the bar, as shown in the example on the right in Figure 4.

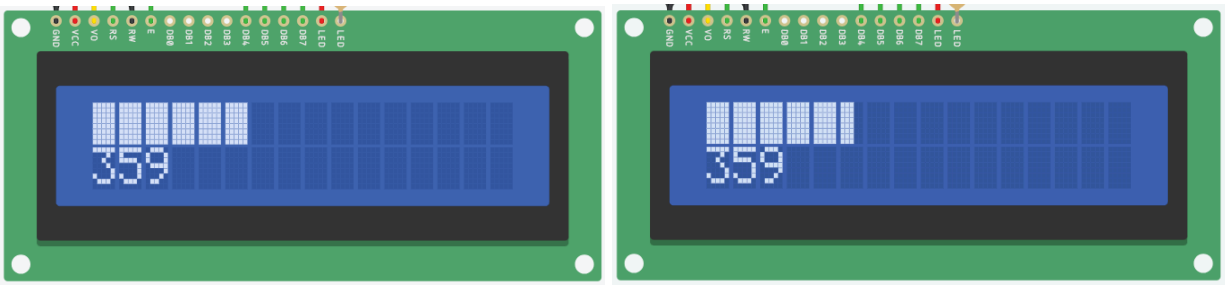

Figure 4. Example of printed value and bar (optional objective on the right - higher precision in the bar).